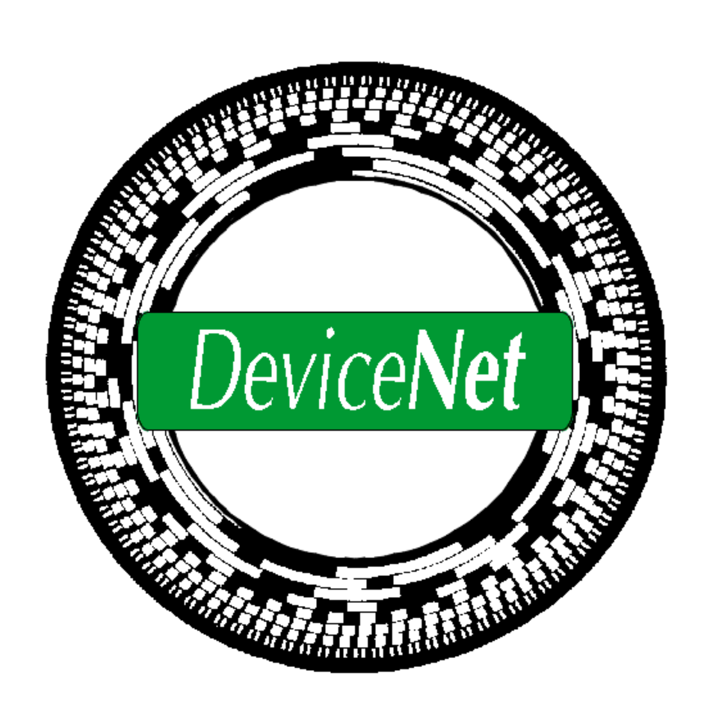
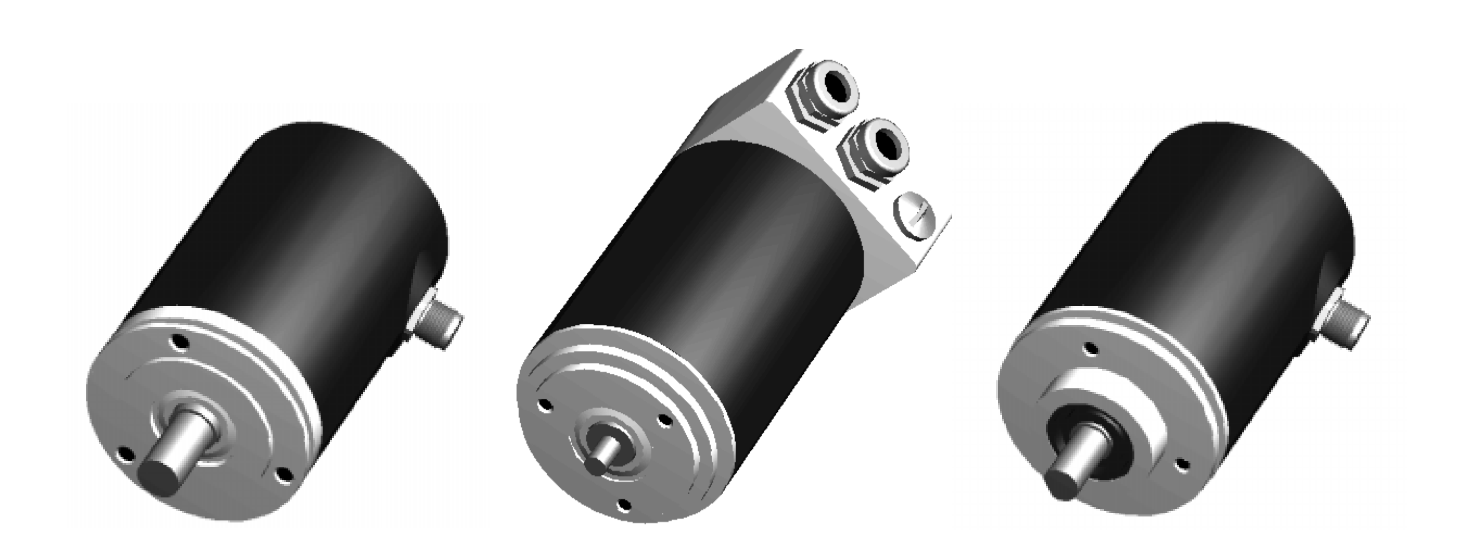


Manuel d'utilisation



POB 10 50 63 ■ 40041 Düsseldorf ■ Germany
Tel. +49/211/63 20 67 ■ Fax +49/211/63 77 05
e-mail: info@twk.de ■ internet: www.twk.de



Sommaire



1. Introduction

Device Net est un système de bus de terrain basé sur le réseau CAN (Controler Area Network). Le bus CAN a été créé par la société allemande Robert Bosch et développé spécialement pour l'industrie automobile. Il est utilisé aujourd'hui dans de nombreux modèles de véhicules de classe supérieure.

Impliquées par les hautes exigences de l'industrie automobile, les caractéristiques du réseau CAN, valables également pour Device Net, sont les suivantes :

- grande sécurité de transmission (distance Hamming = 6)
- temps de réponse courts
- format des données : 8 octets maximum
- système de bus de terrain basé sur la transmission de messages
- disponibilité de composants éprouvés et bon marché
- concept ouvert et facilité d'extension

Alors que le protocole CAN, normalisé ISO/DIS 11898, ne décrit que les couches 1 et 2 des modèles de communication ISO/OSI, DeviceNet, lui, spécifie de plus le médium de transmission de la couche 1 et la couche 7.

Le tableau suivant est explicite :

Couche ISO 7	Application Layer	Spécification DeviceNet
Couche ISO 2	Data Link Layer (Protocol Layer)	Spécification du protocole CAN
Couche ISO 1	Physical Signaling	
	Transceiver	Spécification DeviceNet
	Transmission Media	

Illustration 1.1 Modèle de communication DeviceNet

C'est justement la disponibilité de composants bon marché et la grande sécurité du réseau CAN qui le rendent très intéressant pour les techniques d'automatisation. L'organisme européen pour le développement de CAN est le CiA - CAN in Automation (http://www.can-cia.de). Cet organisme regroupant fabricants et utilisateurs représente aussi bien le protocole CANopen développé en Allemagne, que le protocole DeviceNet promu plus particulièrement par Allen-Bradley, que d'autres dérivés.

Le principal organisme international regroupant fabricants et utilisateurs de DeviceNet est cependant le ODVA - Open DeviceNet Vendor Association (http://www.odva.org). Il est l'éditeur de la spécification DeviceNet |1|.

La spécification par DeviceNet de la couche 7 et du hardware de la couche 1 entraînent les propriétés suivantes :

- possibilité de connexion avec trunkline et dropline
- jusqu'à 64 participants
- possibilité de retirer un participant du bus sans en interrompre son fonctionnement
- transfert des données et alimentation des participants par l'intermédiaire d'un seul câble
- vitesse de transmission : 125, 250, 500 kBaud (voir tableau 1.1)

vitesse de transmission	longueur max. trunkline *	longueur max.	
		1 dropline	totale
125K	500 m	6 m	156 m
250K	250 m	6 m	78 m
500K	100 m	6 m	39 m

Tableau 1.1 Longueurs des câbles

* Ces valeurs sont valables pour le câble épais (DeviceNet thick cable). Pour le câble fin (DeviceNet thin cable) la longueur maximale est toujours de 100 m et indépendante de la vitesse de transmission.

- protection contre les erreurs de câblage
- support des participants alimentés par le câble du bus et de ceux qui possèdent une alimentation propre
- couche d'application basée sur un système d'objets
- support de la communication entre un nombre illimité de participants et la communication maîtres/esclaves
- détection d'adresse attribuée deux fois

2. Schéma de principe : codeur TWK pour DeviceNet

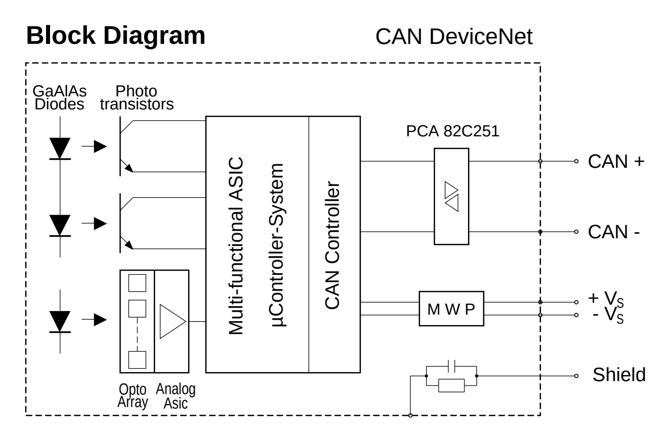


Illustration 2.1 Schéma de principe du codeur CRN/D

L'information concernant la position actuelle du codeur est saisie grâce à un ASIC multifonction et à un microprocesseur intégré (Hostcontroller). La communication entre le Hostcontroller et le réseau CAN ainsi que l'implémentation du protocole de communication ont lieu grâce au CAN-Controller SJA1000. Ce contrôleur contient par exemple la priorité de l'accès au bus définie par les identificateurs des messages, le routines de traîtement des erreurs, les routines des pertes d'arbitrage lors de l'accès au bus, etc.

L'interface entre le CAN-Controller et le bus est réalisée par le CAN Transceiver 82C251 selon le CiA Standard ISO/DIS 11898. D'après la spécification DeviceNet il est prévu une MWP (Mis-Wiring Protection) - commande - pour l'alimentation et une combinaison RC pour le blindage.

Dans DeviceNet la résistance de clôture n'est jamais interne au participant, elle doit toujours être branchée spécialement à l'extrémité de la Trunkline.

3. Remarques concernant l'installation

3.1 Codeur avec connecteur

Ce codeur est pourvu d'un connecteur Micro ou Mini à 5 broches (pour les raccordements voir supplément TY...). L'adressage et le réglage des vitesses de transmission s'effectuent via Software (par ex. DeviceNet-Manager d'Allen-Bradley). (voir aussi chapitre 9.5)

Le réglage d'usine est : adresse 1
vitesse de transmission 125 KB

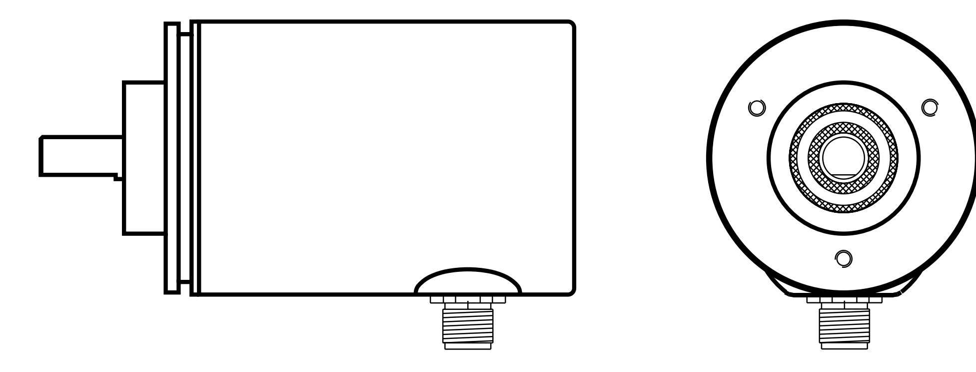


Illustration 3.1 Codeur avec connecteur

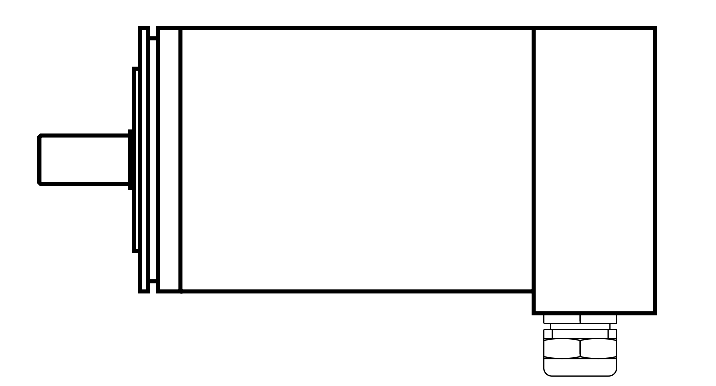


3.2 Codeur avec boîtier de raccordement

Le modèle de codeur avec boîtier de raccordement est pourvu d'un presse-étoupe PG pour le Bus-In et d'un pour le Bus-Out. Le bus et l'alimentation, rassemblés dans un même câble, sont relies à l'intérieur du boîtier sur des borniers de raccordement. (voir le plan de connexion TY... joint au boîtier).

C'est à l'intérieur du boîtier que s'effectuent l'adressage (MAC-ID) et le réglage de la vitesse de transmission.

Le codeur est relié au boîtier grâce à un connecteur Sub-D à 15 broches. En cas de disfonctionnement du codeur celui-ci peut être échangé sans problème ; 2 vis fixent le boîtier au codeur.



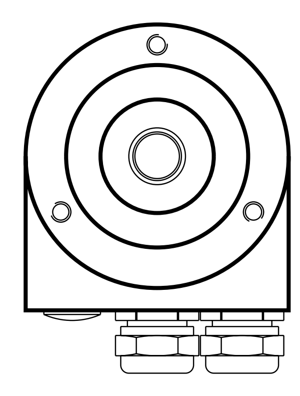


Illustration 3.2 Codeur avec boîtier de raccordement

3.2.1 Adressage (MAC-ID)

Le MAC-ID (Media Access Control Identifier) peut être réglé via le commutateur Dip (valeurs comprises entre 0 et 63) selon le tableau 3.1. La valeur par défaut est 1.

commutateur	6	5	4	3	2	1	adresse
valeur	2^5	2^4	2^3	2^2	2^1	2^0	
	0	0	0	0	0	0	0
	0	0	0	0	0	1	1
ON = 1	...						...
OFF = 0	...						...
	1	1	1	1	1	1	63

Tableau 3.1 Adressage (MAC-ID)

3.2.2 Réglage de la vitesse de transmission

commutateur	8	7	vitesse de transmission
	0	0	125KB
ON = 1	0	1	250KB
OFF = 0	1	0	500KB
	1	1	500KB

Tableau 3.2 Réglage de la vitesse de transmission

Lors de ce réglage, il faut prendre en considération les longueurs de câble autorisées (tableau 1.1).



3.2.3 LED d'état

Dans le boîtier se trouvent 3 LEDs qui renseignent sur l'état du codeur. Une LED verte est le témoin de l'alimentation (V), une LED rouge et une verte (MNS) forment ensemble la LED d'état module/réseau définie par la spécification DeviceNet.

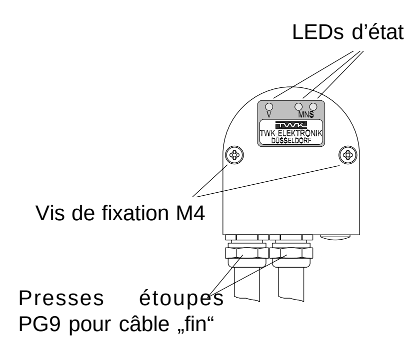


Illustration 3.1 Boîtier de raccordement

LED MNS	Etat	Explication
éteint	pas branché, offline	l'appareil est offline : - Dupl. MAC-ID check pas terminé - pas d'alimentation
clignotant vert	appareil actif et online, pas de connexion	l'appareil fonctionne normalement et est online, il n'y a pas de connexion - le codeur n'a pas encore été configuré par le maître - configuration incomplète ou erronée
vert	appareil actif et online, connexions établies	l'appareil fonctionne normalement et est online, les connexions sont établies - le codeur a été configuré par le maître
clignotant rouge	erreur minime et/ou interruption de la connexion	erreur corrigeable et/ou une ou plusieurs connexions I/O sont interrompues
rouge	disfonctionnement de l'appareil ou grave erreur de communi-cation	l'appareil a un disfonctionnement irréparable et doit être remplacé ou l'appareil a constaté une erreur qui empêche la communication avec le réseau - adresse (MAC-ID) attribuée deux fois
clignotant rouge & vert	comm. interrompue et réception d'un indicateur de demande d'erreur de communication	erreur de communication particulière. L'appareil a décelé une erreur dans l'accès au réseau et est en mode erreur de communication

Tableau 3.3 LED d'état module/réseau

4. Les modes de fonctionnement du codeur

Les modes de fonctionnement du codeur déterminent le déclenchement de la saisie de la valeur instantanée de la position. L'utilisateur a le choix entre 3 modes de fonctionnement :

1. Polling Mode
2. Bit Strobed Mode
3. Change of State Mode
4. Cyclic Mode

Il est également possible d'actionner plusieurs modes simultanément.

4.1 Polling Mode

Le mode de fonctionnement standard du système maître/esclave est le Polling Mode. Dans ce cas de figure le maître interroge de façon cyclique tous les participants. Dans un *Scan* toutes les données de sorties sont transmises aux esclaves et toutes les données d'entrées sont transmises par les esclaves. En général le temps entre 2 Scans est réglable dans le maître.

Le codeur transmet ses données d'entrée sur demande cyclique (Poll-Command) du maître.

4.2 Bit Strobed Mode

L'utilisateur désire-t-il s'adresser à un esclave (ou à plusieurs esclaves) en particulier ou à tous à un moment donné, alors il doit activer le Bit Strobed Mode.

Quand le codeur fonctionne dans ce mode il envoie ses données d'entrée en réponse à la commande Bit Strobe.

4.3 Change of State Mode

Si un participant ne doit envoyer uniquement ses données d'entrée que lorsque celles-ci ont changé, l'utilisateur active alors le Change of State Mode. Dans ce mode l'utilisateur peut également régler l'**Heartbeat Rate**. Le participant envoie alors les nouvelles données d'entrée, qu'elles aient changé ou non, après l'écoulement d'un laps de temps fixé par lui.

Dans ce mode le codeur envoie donc sa position lorsque celle-ci a changé **et** après écoulement de l'**Heartbeat Rate**.La variable Heartbeat Rate est réglable dans une plage de 2 à 65535 ms par pas de 2 ms. (la valeur maxi. que peut prendre l'Heartbeat Rate du DeviceNet-Manager de chez Allen-Bradley est seulement de 48 ms).

De plus il est possible de retarder l'envoi des données dans une plage de 2 à 65535 ms (production inhibit time). La nouvelle valeur instantanée de la position n'est envoyée au maître qu'après l'écoulement du temps de retard.

4.4 Cyclic Mode

Dans ce mode le participant envoie ses données d'entrées à un moment donné, fixé par l'utilisateur. Cette variable **Send Rate** peut être réglée dans chaque participant.

Le Cyclic Mode est donc un Change of State Mode qui envoie les données d'entrée de façon cyclique et non pas lorsque ces dernières ont changé.

En fait ce mode n'est qu'une variante du Change of State Mode. C'est pour cette raison que l'on parle aussi du Change of State Mode / Cyclic Mode.

5. Codeur - paramètres

Les paramètres du codeur permettent une adaptation aux besoins des clients. Ces paramètres peuvent être écrits et lus grâce au transfert de données propre à DeviceNet qu'est l'**Explicit Messaging**.

Une commande „Save" permet de mémoriser les paramètres du codeur pour parer à une coupure de secteur et une commande „Restore" permet de charger les valeurs par défaut.

Attention : après utilisation des commandes Save ou Restore le codeur ne transmet pas de valeur instantanée de la position pendant environ 300 ms. Il ne faut donc les utiliser en aucun cas pendant le fonctionnement d'une machine.

5.1 Description des paramètres

Paramètres	Explication
Evolution du code	Il indique dans quel sens de rotation la valeur du code de sortie augmente CW - valeur croissante pour rotation dans le sens des aiguilles d'une montre (clockwise) CCW - valeur croissante pour rotation dans le sens contraire des aiguilles d'une montre (counter clockwise) Remarque : regard face à l'axe
Calibrage	Active ou désactive le calibrage. Les valeurs pour la résolution, plage de mesure et la valeur de référence ne sont actives que lorsque le calibrage est activé.
Résolution	Elle indique le nombre de pas par tour.
Plage de mesure	Il indique la résolution totale, en la calculant ainsi : résolution totale = nombre de tour x résolution
Valeur de référence	C'est la valeur qui est indiquée au point de référence. Elle peut prendre les valeurs de 0 à résolution totale - 1.
Plage de travail inférieure	valeur limite inférieure du réglage de la plage de travail effectué par un utilisateur.
Plage de travail supérieure	valeur limite supérieure du réglage de la plage de travail effectué par un utilisateur.

Tableau 5.1 Codeur - paramètres

Remarque : La résolution totale doit être choisie de telle manière à ce que le nombre de rotation soit > ou = à 1. Si le codeur est utilisé en fonctionnement continu, le nombre de rotation doit être égal à 2^n (avec n =1,2,..,12).



5.2 Codeur - paramètres

Paramètres	Plage de valeur	Valeur par défaut	Type de donnée
Evolution du code Code sequence	CW/CCW	CW	BOOL
Calibrage Scaling	activé/désactivé	désactivé	BOOL
Résolution Resolution	1 - 8192	8192	UNSIGNED INTEGER
Plage de mesure Measuring Range	1 - 33.554.432	33.554.432	UNSIGNED INTEGER
Valeur de référence Preset Value	0 - (réglage plage de mesure -1)	0	UNSIGNED INTEGER
Plage de travail inférieure Work Area Low Limit	0 - 33.554.432	1.048.575	UNSIGNED INTEGER
Plage de travail supérieure Work Area High Limit	0 - 33.554.432	32.505.856	UNSIGNED INTEGER

Tableau 5.2 Paramètres
(La plage de valeur représentée ainsi que les valeurs par défaut sont valables pour une résolution de 25 Bit.)

Remarque : pour la présentation des paramètres dans le protocole le Low-Byte précède toujours le High-Byte.

6. Codeur - Données d’entrée

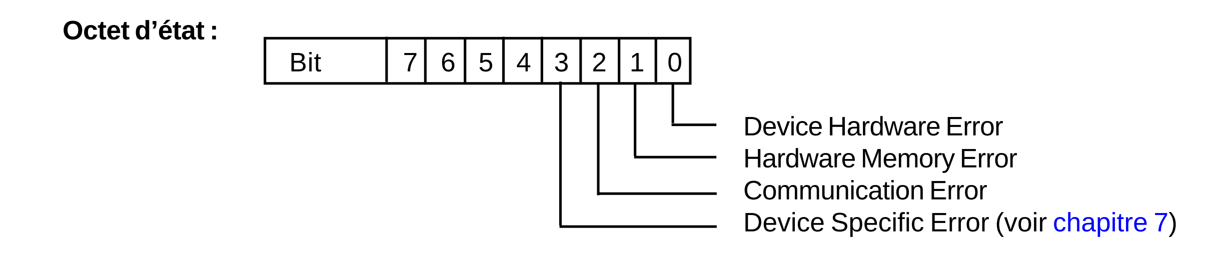
Dans le transfert de données I/O - **I/O-Messaging** - le codeur transmet comme données d’entrée (vues du côté du maître) la valeur instantanée de la position et un octet d’état. Le mode de fonctionnement du codeur détermine le temps de déclenchement de la saisie de la valeur instantanée de la position. C’est ainsi que le codeur transmet 5 octets de données d’entrée qui doivent être interprétées de la sorte :

	Champ de données				
Octet	1	2	3	4	5
Bit	7 - 0	15 - 8	23 - 16	31 - 24	39 - 32
Valeur	$2^7 - 2^0$	$2^{15} - 2^8$	$2^{23} - 2^{16}$	$2^{31} - 2^{24}$	0000XXXX
Signification	Valeur instantanée de la position				Octet d’état

Tableau 6.1 Données d’entrée

0 = non utilisé , X = utilisé

Octet d’état :



7. Codeur - informations d'état

En plus des données d'entrée du I/O-Messaging, le codeur transmet d'autres informations d'état qui, comme les paramètres, peuvent être lues par l'**Explicit Messaging**. Les données d'entrée (chapitre 6) peuvent être également lues par l'Explicit Messaging.

Désignation	Type de donnée	Explication
Plage de travail registre d'etat Area state register	Unsigned short integer	Bit 0 : non utilisé Bit 1 : égal à 1 si valeur de position > plage de travail sup. Bit 2 : égal à 1 si valeur de position < plage de travail inf. Bit 3-7 : non utilisés
Mode de fonction. Operating Status	Word	Bit 0 : 0 = CW; 1 = CCW Bit 2 : 0 = calibrage désactivé ; 1 = calibrage activé
Résolution max. par tour Single Turn Resolution	Unsigned integer	Indication en pas/tour
Nombre max. de tour Number of distinguishable revolutions	Unsigned integer	Indication en tour
Alarmes Alarms	Word	Bit 0 : 1 = Position Error Bit 1-11 : non utilisés Bit 12 : 1 = Eeprom Error Bit 13 : 1 = CRC error Bit 14,15 : non utilisés
Alarmes supportées Supported alarms	Word	Indique quelles annonces d'alarmes énumérées par Word Alarms sont supportées par le codeur (A ce moment précis toutes les alarmes sont supportées)
Mises en garde Warnings	Word	Bit 0-4 : non utilisés Bit 5 : 1 = valeur inst. de la position différente de la valeur de référence Bit 6-15 : non utilisés
Mises en garde supportées Supported Warnings	Word	Indique quelles annonces de mise en garde énumérées par Word Warnings sont supportées par le codeur (A ce moment précis toutes les mises en garde sont supportées)
Versions Profile et Software Profile and Software version	DWord	Word 0 = version Profile (représentation hexadécimale) Word 1 = version Software (représentation hexadécimale)

Tableau 7.1 Informations d'état

Remarque : pour la présentation des paramètres dans le protocole, le Low-Byte précède toujours le High-Byte.



8. Le protocole DeviceNet de la couche 7

Remarque: pour raccorder le codeur à un automate d'Allen-Bradley à l'aide du DeviceNet-Manager ou du RS-Networx la connaissance de ce chapitre n'est pas vraiment nécessaire. Il est donc possible de poursuivre directement la lecture de ce manuel avec les chapitres 9 ou 10.

8.1 Structure des objets de DeviceNet

La couche 7 du modèle de communication ISO/OSI de DeviceNet est composée d'objets. Chaque participant se compose d'un certain nombre d'objets. Chaque objet comprend des attributs (données) et des services (fonctions) d'un composant bien particulier du participant. Les objets qui représentent les mêmes composants du système sont regroupés en classes.

L'illustration ci-dessous montre de façon très abstraite la structure de DeviceNet du point de vue de l'objet :

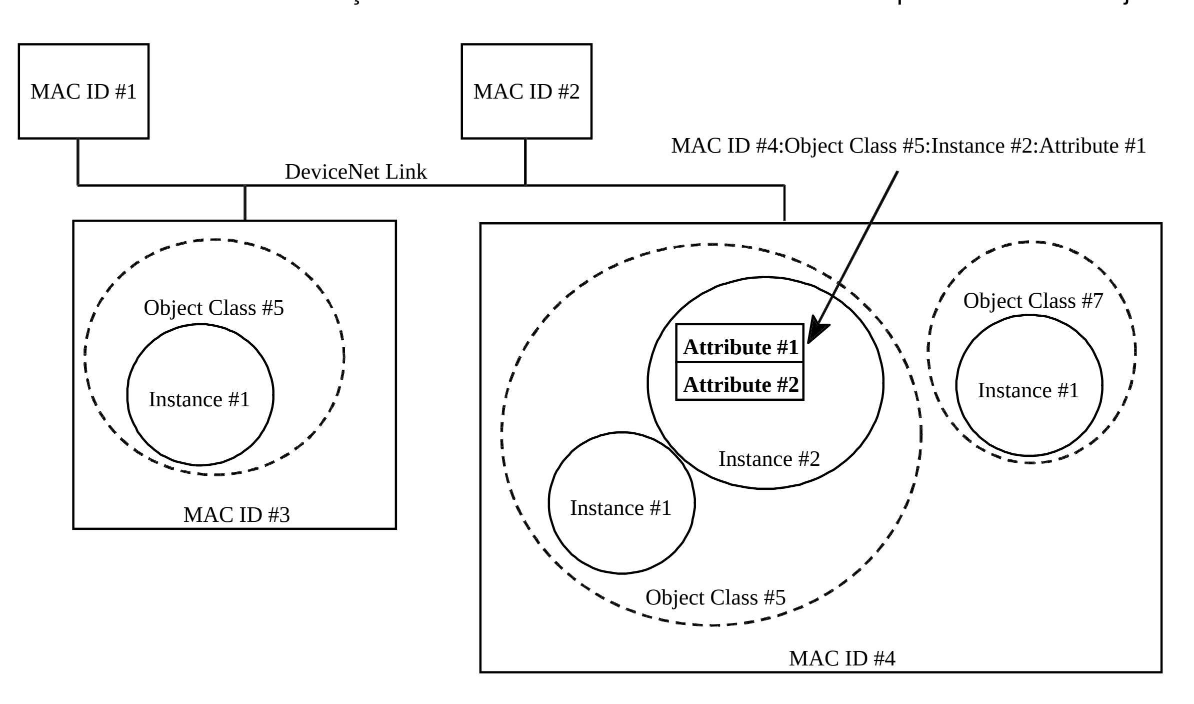


Illustration 8.1 Structure des objects de DeviceNet

Chaque objet rempli par des données de l'utilisateur (instanced) est nommé instance. Un objet peut avoir ainsi plusieurs instances.

Toutes les classes, instances, attributs et services reçoivent des identificateurs (valeur Integer). Avec ceux-ci et l'adresse des participants (MAC-ID) il est possible de s'adresser à tous les attributs et services du bus. (L'illustration ci-dessus représente l'adressage d'un seul attribut)

8.2 Les objets du codeur

Le **Predefined-Master-Slave Connection Set** est une présentation simplifiée des multiples possibilités de DeviceNet. Il représente un sous-ensemble de DeviceNet et offre toutes les fonctions nécessaires aux relations maître/esclave aujourd'hui d'usage. Ce sont là essentiellement les modes de connexion (modes de fonctionnement) qui sont fixés.

En outre c'est dans le Predefined-Master-Slave-Connection-Set que le **Group 2 Server/Client** est distingué du **Group 2 only Server/Client**. Ce dernier se limite aux modes de connexion pour les simples participants I/O, et rend ainsi possible des créations d'esclaves DeviceNet bon marché. La spécification DeviceNet |1| donne à ce sujet davantage d'informations. Toutes les explications qui suivent se réfèrent à l'implémentation DeviceNet du codeur.

Le codeur représente un **Group 2 only Server**.



Les objets ci-dessous sont contenus dans le codeur :

Class ID	Class	Explication
01_{hex}	Identity Object	contient des informations générales sur l'esclave, comme par ex. : ID-fabricant, numéro de série, etc.
02_{hex}	Message Router Object	reçoit tous les messages et les transmet aux objets correspondants
03_{hex}	DeviceNet Object	contient la configuration et l'état de la connexion, par ex. : MAC-ID, vitesse de transmission
04_{hex}	Assembly Object	permet le rassemblement des attributs d'objets différents dans un même objet
05_{hex}	Connection Object	gère aussi bien les connexions I/O que celles de l'Explicit Messaging. Une instance de cet objet est créée pour chaque connexion (Polling, Bit-Strobe, Explicit Messaging par ex.)
23_{hex}	Position Sensor Object	contient toutes les données du codeur (données d'entrée, paramètres et information d'état)
$2b_{hex}$	Acknowledge Handler Object	surveille la réception d'informations Acknowledge pour les objets producteurs d'informations (connexion COS par ex.)

Tableau 8.1 Objets du codeur

Le schéma ci-dessous illustre les interfaces de chaque objet :

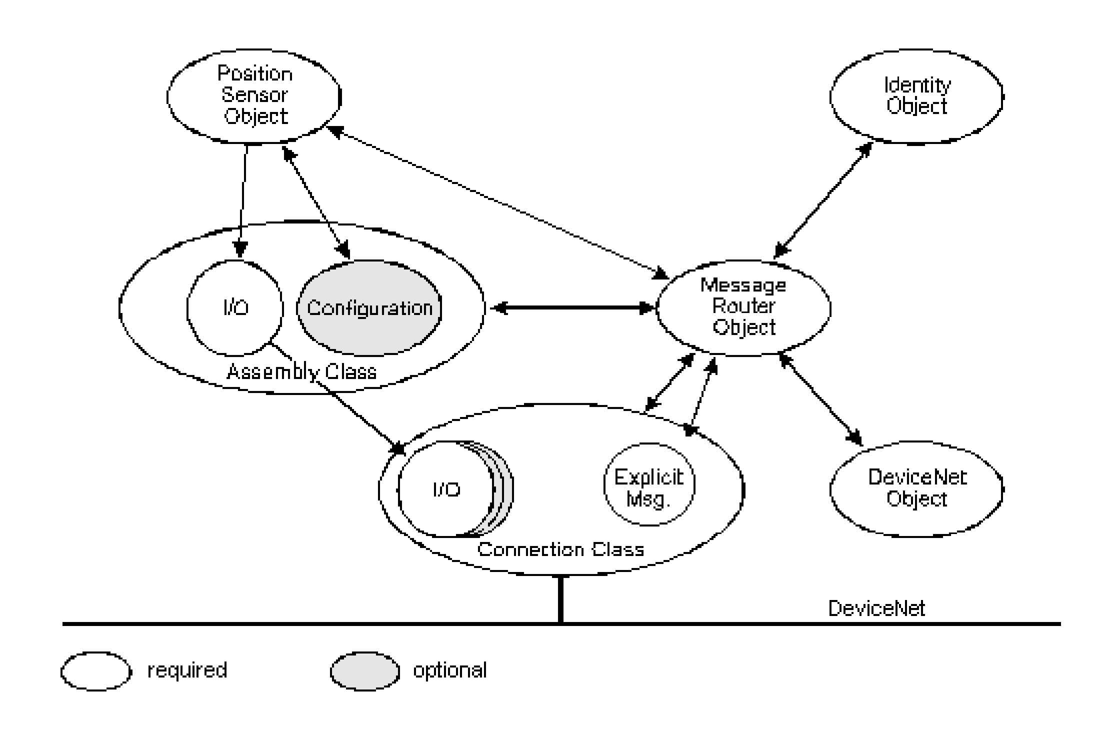


Illustration 8.2 Interfaces des objets



Comme en témoigne également le schéma, tous les objets (excepté le Connection-Object) n’ont qu’une instance. Pour le Connection-Object une instance est créée pour chaque connexion, Poll-Mode ou Explicit Messaging par ex.. Les instances ID’s suivantes sont attribuées aux différents types de connexion :

Connection Instance ID #	Type de connexion
1	Explicit Messaging
2	Poll-Mode
3	Bit-Strobe-Mode
4	Change-of-State-Mode

Tableau 8.2 Instances du Connection-Object

8.2.1 Identity Object

Class Code 01_{hex}

Class Attributes

non supportés

Instance Attributes

Attr. Id	Attributes	Accès	Type de données DeviceNet	Remarque
1	Vendor ID	lecture	UINT	TWK = 407
2	Device Type	lecture	UINT	TWK = generic
3	Product Code	lecture	UINT	0x01
4	Revision	lecture	UINT	1.1
5	Status	lecture	WORD	
6	Serial Number	lecture	UDINT	
7	Product Name	lecture	SHORT-STRING	Encoder CRN/D

Services

Service Code	Service Name	Remarque
$0E_{hex}$	Get_Attribute_Single (lecture)	remet la valeur de l’attribut
05_{hex}	Reset	

8.2.2 Message Router Object

Class Code 02_{hex}

Class Attributes

non supportés

Instance Attributes

non supportés

Services

non supportés

8.2.3 DeviceNet Object

Class Code 03_{hex}

Class Attributes

Attr. Id	Attribute	Accès	Type de données DeviceNet	Remarque
1	Revision	lecture	UINT	Revision = 002

Instance Attributes

Attr. Id	Attribute	Accès	Type de données DeviceNet	Remarque
1	MAC ID	lecture/écriture*	USINT	Revision = 002
2	Baud Rate	lecture/écriture*	USINT	Range 0-2
3	BOI	lecture	BOOL	Value = 0
4	Bus-Off Counter	lecture/écriture	USINT	
5	Allocation Information	lecture	STRUCT of: BYTE USINT	Allocation Choice Byte Master's MAC ID

* seulement sur le modèle de codeur avec connecteur

Services

Service Code	Service Name	Remarque
$0E_{hex}$	Get_Attribute_Single (lecture)	remet la valeur d'un attribut
10_{hex}	Set_Attribute_Single (écriture)	modifie la valeur d'un attribut
$4B_{hex}$	Allocate_Master/Slave_Connection_ Set	Predefined Master/Slave Connection Set est demandé
$4C_{hex}$	Release_Group_2_Identifier_Set	les connexions via Master/Slave sont interrompues

8.2.4 Assembly Object

Class Code 04_{hex}

Class Attributes

pas supportés

Instance Attributes

Attr. Id	Attribute	Accès	Type de données DeviceNet	Remarque
3	Data	lecture/écriture	ARRAY	



Services

Service Code	Service Name	Remarque
0E$_{hex}$	Get_Attribute_Single (lecture)	remet la valeur d’un attribut
10$_{hex}$	Set_Attribute_Single (schreiben)	modifie la valeur d’un attribut

8.2.5 Connection Object

Class Code 05$_{hex}$

Class Attributes

non supportés

Instance Attributes

Attr.Id	Attribute	Accès	Type de données DeviceNet	Remarque
1	state	lecture	USINT	
2	instance_Type	lecture	USINT	
3	transportClass_trigger	lecture	BYTE	
4	produced_connection_id	lecture	UINT	
5	consumed_connection_id	lecture	UINT	
6	initial_comm_characteristics	lecture	BYTE	
7	produced_connection_size	lecture	UINT	
8	consumed_connection_size	lecture	UINT	
9	expected_packet_rate	lecture/ écriture	UINT	En fixant cette valeur la connexion est „établie“ et tous les autres paramètres sont acceptés
12	watchdog_timeout_action	lecture	USINT	
13	produced_connection_path_length	lecture	UINT	
14	produced_connection_path	lecture	Array of UINT	
15	consumed_connection_path_length	lecture	UINT	
16	consumed_connection_path	lecture	Array of UINT	
17	production_inhibit_time	lecture/ écriture	UINT	

Services

Service Code	Service Name	Remarque
0E$_{hex}$	Get_Attribute_Single (lecture)	remet la valeur d’un attribut
10$_{hex}$	Set_Attribute_Single (écriture)	modifie la valeur d’un attribut



8.2.6 Position Sensor Object

Class Code 23_{hex}

Class Attributes

Attr. Id	Attribute	Accès	Type de données	Remarque
1	Revision	lecture	UINT	
2	Max. Instance	lecture	UINT	

Instance Attributes

Partie spécifique à l'ODVA :

Attr.Id	Attribute	Accès	Type de donnée	Remarque
1	# of Attributes	lecture	USINT	nombre d'attributs d'une instance
2	Attributes	lecture	Array of/ USINT	tous les Attribute ID's d'une instance présents
3	Value	lecture	UDINT	position actuelle du codeur
11	Value Direction Control	lecture/ écriture	BOOL	0 = cw, 1 = ccw
39	Status Byte	lecture	USINT	Bit 0: 1=hardware error Bit 1: 1=memory error Bit 2: 1=communication error Bit 3: 1=device specific error

Partie spécifique au fabricant :

112	Scaling	lecture/ écriture	BOOL	0=scaling disable;1=scaling enable
113	Measuring units per revolution	lecture/ écriture	UDINT	max. 8192 steps/U
114	Total measuring range in measuring units	lecture/ écriture	UDINT	max. 3554432 steps
115	Preset Value	lecture/ écriture	UDINT	résolution totale max. - 1
128	Area state register	lecture	USINT	Bit 1: 0=OK; 1= supérieur à limite max. Bit 2: 0=OK; 1= inférieur à limite min.
129	Work area low limit	lecture/ écriture	UDINT	default = 1048575 steps (pour résolution 25 Bit)
130	Work area high limit	lecture/ écriture	UDINT	default = 32505856 steps (pour résolution 25 Bit)
144	Operating status	lecture	WORD	Bit 0: 0=cw; 1=ccw Bit 2: 0=scaling disable;1=scaling enable
145	Single Turn Resolution	lecture	UINT	8192 steps/U (13 Bit)
146	Number of distinguish-able revolutions	lecture	UINT	4096 U (12 Bit)
147	Alarms	lecture	WORD	Bit 12: 1 = Eeprom error Bit 13: 1 = CRC error Bit 14: 1 = XRAM error
148	Supported alarms	lecture	WORD	Toutes les alarmes sont supportées
149	Warnings	lecture	WORD	Bit 5: 1=valeur inst. différente de la valeur de référence
150	Supported warnings	lecture	WORD	Toutes les mises en garde sont supportées
151	Profile and software version	lecture	DWORD	Word 0 = version Profile Word 1 = version Software
153	Offset Value	lecture	UDINT	non supportée



Services

Service Code	Service Name	Remarque
$0E_{hex}$	Get_Attribute_Single (lecture)	remet la valeur d'un attribut
10_{hex}	Set_Attribute_Single (écriture)	modifie la valeur d'un attribut
15_{hex}	Restore	inscrit sur l'EEPROM les valeurs par défaut
16_{hex}	Save	inscrit sur l'EEPROM les attributs non volatiles

8.3 Connexions DeviceNet

DeviceNet est un réseau basé sur les connexions. Chaque communication a lieu via des connexions (canaux). Avant qu'une communication puisse avoir lieu, il faut qu'une connexion soit créée. Pour cette connexion les ressources correspondantes doivent être mises en place et les identificateurs déterminés.

Le champ d'Identifier se compose comme celui de CAN de 11 Bits. DeviceNet divise cet espace des adresses en 4 domaines Group 1,Group 2, Group 3 et Group 4 :

IDENTIFIER BITS											HEX RANGE	IDENTITY USAGE
10	9	8	7	6	5	4	3	2	1	0		
0	Group 1 Message ID				Source MAC ID						000 - 3ff	Message Group 1
1	0	MAC ID						Group 2 Message ID			400 - 5ff	Message Group 2
1	1	Group 3 Message ID			Source MAC ID						600 - 7bf	Message Group 3
1	1	1	1	1	Group 4 Message ID (0 - 2f)						7c0 - 7ef	Message Group 4
1	1	1	1	1	1	1	X	X	X	X	7f0 - 7ff	Invalid CAN Identifiers
10	9	8	7	6	5	4	3	2	1	0		

Tableau 8.3 Identificateurs DeviceNet

Dans le Predefined-Master-Slave-Connection-Set seuls les Message Group 1 et Group 2 sont utilisés. Les Message-ID's ont déjà été fixés et n'ont plus besoin d'être négociés. L'attribution des Message-ID's aux connexions se fait de la façon suivante :

IDENTIFIER BITS											IDENTITY USAGE	HEX RANGE
10	9	8	7	6	5	4	3	2	1	0		
0	Group 1 Message ID				Source MAC ID						Group 1 Messages	000 - 3ff
0	1	1	0	1	Source MAC ID						Slave' I/O Change of State or Cyclic Message	
0	1	1	1	0	Source MAC ID						Slave's I/O Bit-Strobe Response Message	
0	1	1	1	1	Source MAC ID						Slave's Poll Response or Change of State/Cyclic Acknow ledge	
1	0	MAC ID						Group 2 Message ID			Group 2 Message	400 - 5ff
1	0	Source MAC ID						0	0	0	Master's I/O Bit-Strobe Command Message	
1	0	Source MAC ID						0	0	1	Reserved for Master's Use - Use is TBD	
1	0	Destination MAC ID						0	1	0	Master's Change of State or Cyclic Acknow ledge Message	
1	0	Source MAC ID						0	1	1	Slave's Explicit/Unconnected Response Messages	
1	0	Destination MAC ID						1	0	0	Master's Explicit Request Messages	
1	0	Destination MAC ID						1	0	1	Master's I/O Poll Command/Change of State/Cyclic Message	
1	0	Destination MAC ID						1	1	0	Group 2 Only Unconnected Explicit Request Messages (reserved)	
1	0	Destination MAC ID						1	1	1	Duplicate MAC ID Check Message	

Tableau 8.4 Identificateurs du Predefined Master/Slave Connection Set



8.4 Le protocole DeviceNet

En principe DeviceNet distingue 2 sortes de protocoles :

- **Explicit Messaging**
- **I/O Messaging**

L'Explicit Messaging est utilisé pour lire ou écrire des attributs de façon ciblée. Il faut également distinguer le protocole **fragmenté** du **non fragmenté**.

L'I/O-Messaging est utilisé pour le transfert rapide de données entrée/sortie. L'utilisateur a à disposition les 8 octets du protocole CAN, ce qui n'est pas le cas pour l'Explicit Messaging.

Dans le codeur, les paramètres et les informations d'état sont lus ou écrits via l'Explicit Messaging et les 5 octets de données d'entrée sont lus via l'I/O-Messaging.

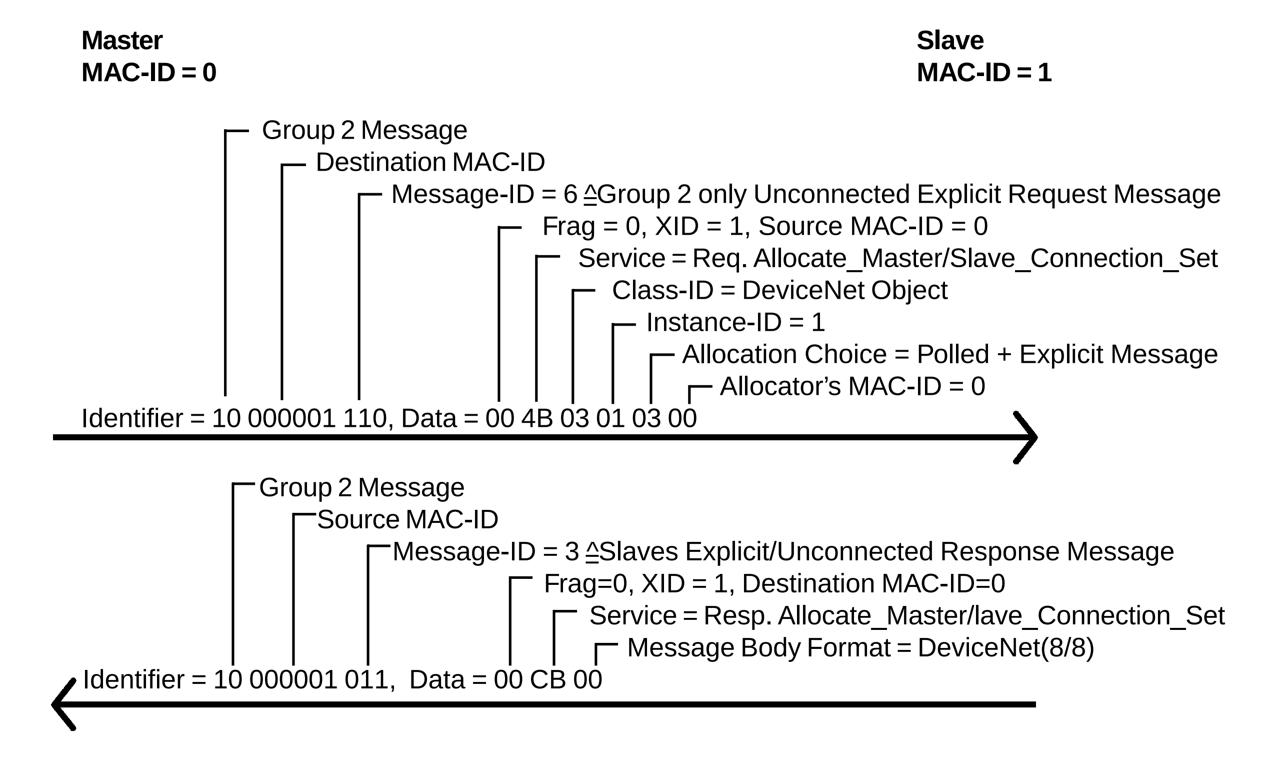
8.5 Réalisation d'une connexion avec le codeur

Chaque connexion avec un esclave est réalisée via le Group 2 only unconnected Explicit Request Message (Group 2 Message-ID 6). Dans le Predefined-Master-Slave-Connection-Set une instance du Connection Object (voir Tableau 8.2) est créée via le service **4B**$_{hex}$ **Allocate_Master/Slave_Connection_Set** du DeviceNet Object. Le mode de connexion (mode de fonctionnement) est donné à ce service dans un **Allocation Choice Byte**.

Allocation Choice Byte

7	6	5	4	3	2	1	0
Reserved	Acknowledge Suppression	Cycle	Change of State	Reserved	Bit Strobed	Polled	Explicit Message

Exemple : installation de la connexion I/O Poll-Mode et de l'Explicit Messaging

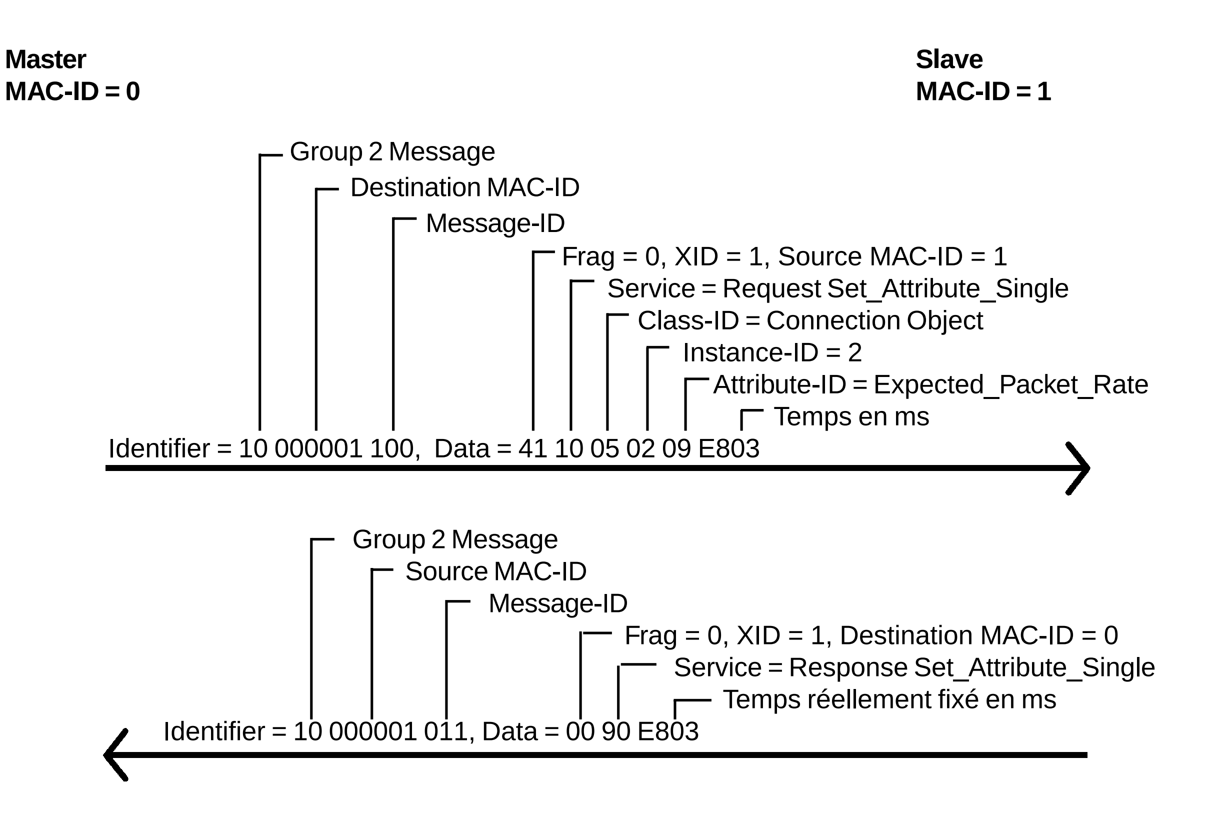


Dans l'exemple ci-dessus une instance est créée avec l'ID1 (Explicit Messaging Connection) et avec l'ID 2 (I/O-Messaging Poll Mode) du Connection Object. Les connexions ainsi établies se trouvent ensuite en mode „Configuring" (Instance-Attribute 1 = 1). Pour qu'elles soient en mode „Established" (Instance-Attribute 1 = 3) il faut fixer la valeur de l'**Expected_Packet_Rate** (Instance-Attribute 9). L'exemple ci-après concerne l'instance Poll-Mode.



Exemple : fixer la valeur de l'Expected_Packet_Rate d'un attribut

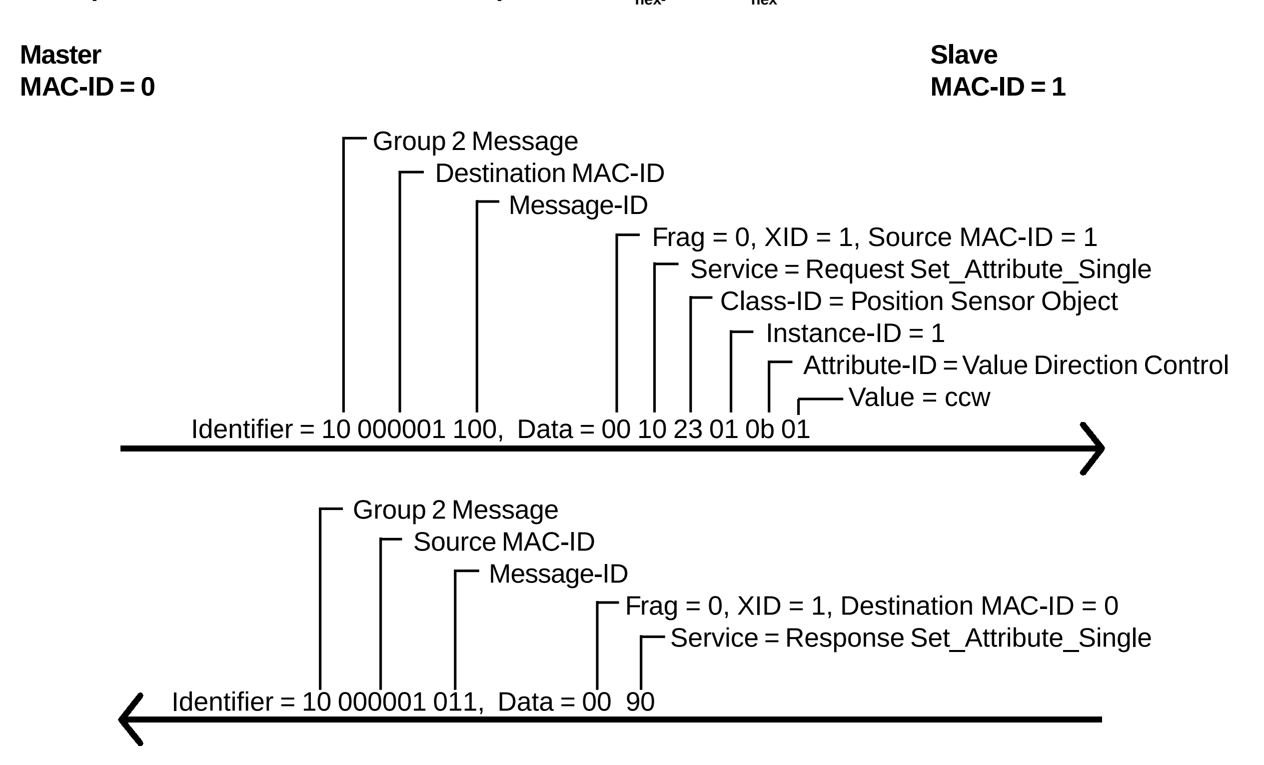
Fixer la valeur de l'attribut d'une instance a lieu de façon unitaire pour tous les objets grâce au service **10$_{hex}$ Set_Attribute_Single**. Dans cet exemple le temps fixé est d'1 seconde.



8.6 Paramétrage du codeur

Le paramétrage du codeur s'effectue de la même manière que celui de la valeur de l'Expected_Packet_Rate avec à un Explicit Message et le service **10$_{hex}$ Set_Attribute_Single** du Position Sensor Object.

Exemple : fixer de l'évolution du code (Attribute b$_{hex}$) auf 01$_{hex}$



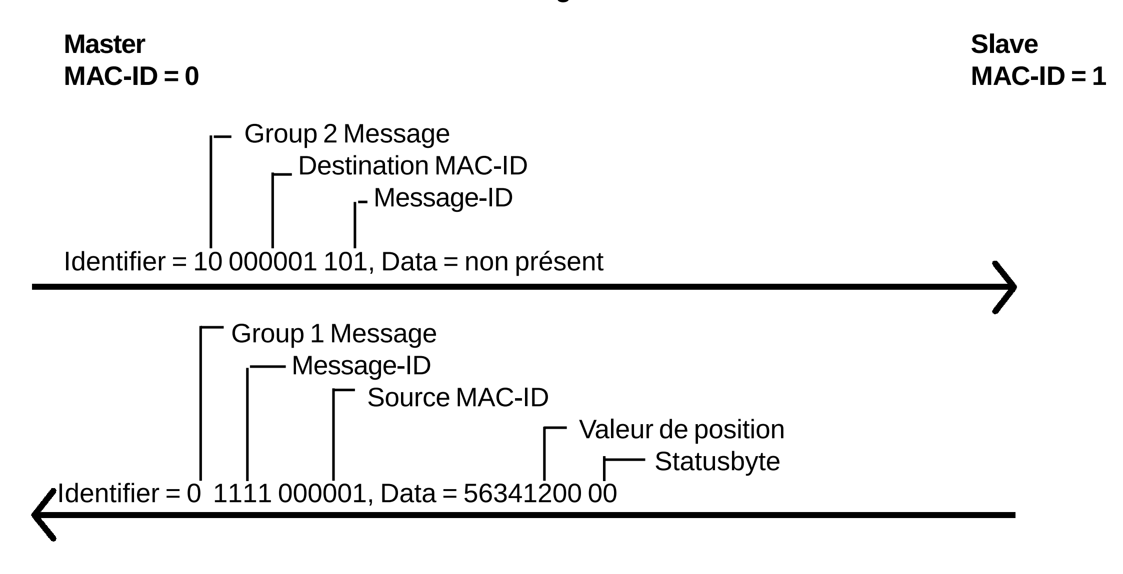


8.7 Interrogation des données d'entrée du codeur

L'interrogation des données instantanées a lieu via l'I/O-Messaging. Selon la connexion installée il faut utiliser un Message ID différent pour le Request.

Lors du Polling, chaque participant est interrogé séparément via un Poll-Command. Les participants qui envoient des données reçoivent également grâce au Poll-Command les données de sortie. Ces dernières n'existent pas dans le codeur.

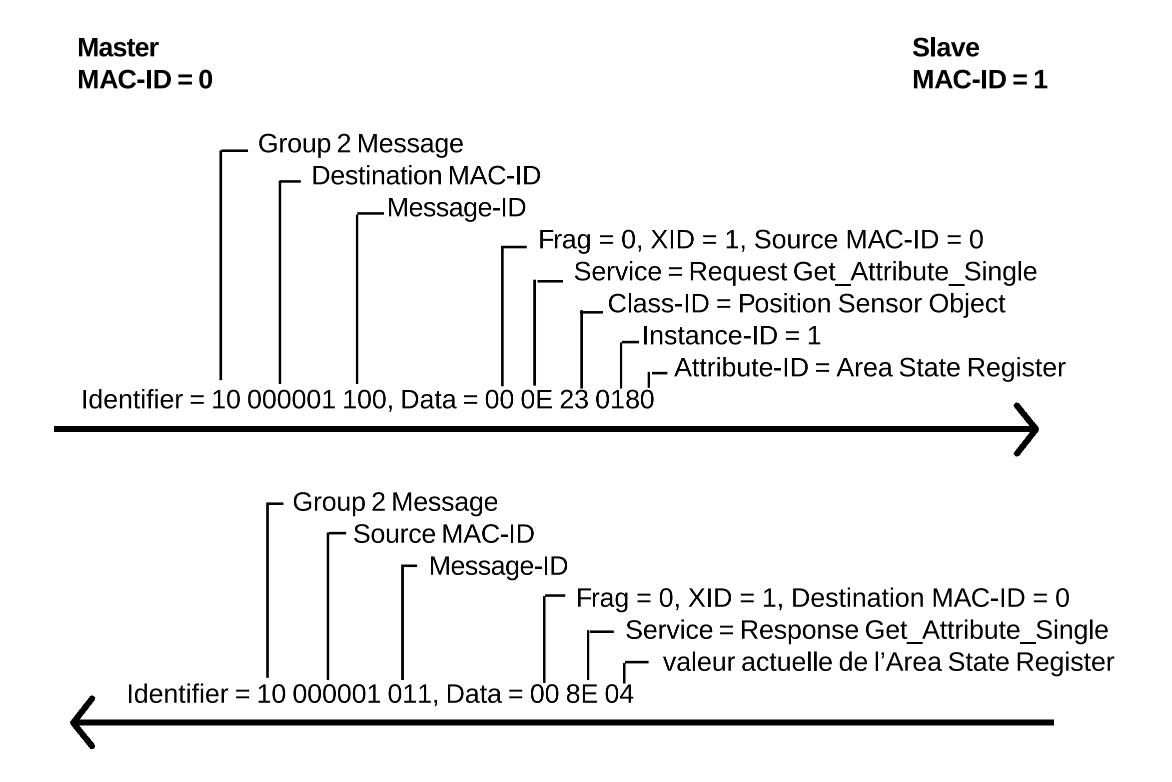
Exemple : lecture des valeurs instantanées en Polling Mode



8.8 Demande d'informations d'état

Les informations d'état du codeur accompagnants les octets d'état des données d'entrée doivent être lues via l'Explicit Messaging. La lecture des attributs d'un objet a lieu pour tous les objets grâce au service **0E$_{hex}$Get_Attribute_Single.**

Exemple : lecture de l'Attribut 80$_{hex}$ Area State Register



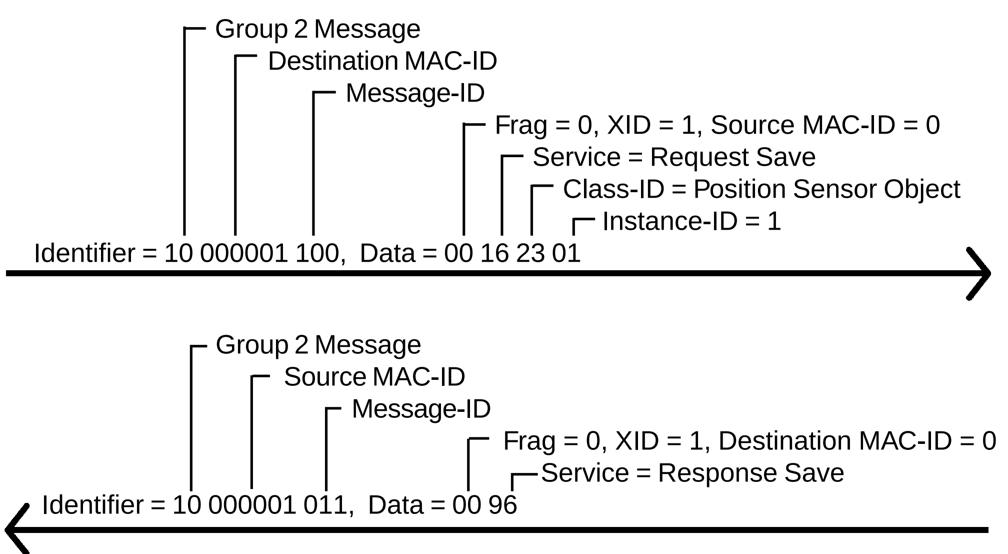


8.9 Enregistrement des valeurs des paramètres dans l'EEPROM

Grâce au service 16_{hex} „Save" du Position-Sensor-Object toutes les valeurs des paramètres sont enregistrées dans l'EEPROM du codeur pour parer à une coupure de secteur.

Exemple :





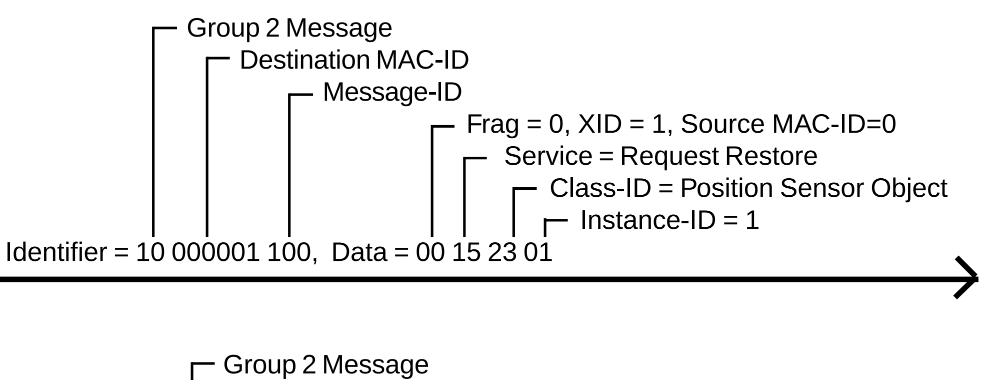
8.10 Chargement des valeurs par défaut des paramètres

Le service 15_{hex} „Restore" charge toutes les paramètres avec leurs valeurs par défaut et les enregistre ensuite dans l'EEPROM. Les valeurs par défaut ne sont actives à nouveau qu'après avoir redémarrer le codeur (tension off/on).

Exemple :

Master
MAC-ID = 0

Slave
MAC-ID = 1



Group 2 Message
Source MAC-ID
Message-ID
Frag = 0, XID = 1, Destination MAC-ID = 0
Service = Response Restore
Identifier = 10 000001 011, Data = 00 95



9. DeviceNet-Manager

Dans ce chapitre est décrit le raccordement du codeur TWK CRN/D au système DeviceNet-Master-Slave d'un automate d'Allen-Bradley, raccordement effectué grâce au DeviceNet-Manager d'Allen-Bradley (Rockwell). La connaissance exacte des objets, thème du chapitre 8, n'est pas nécessaire.

Le système DeviceNet via le DeviceNet-Manager |2| peut être créé de différentes manières. On distingue principalement entre **configuration Offline et configuration Online**.

Lors de la **configuration Offline** il est possible, hors connexion avec le bus, de configurer le bus et de paramétrer les participants. Puis la connexion est établie et les données de l'utilisateur sont transmises au maître et aux participants.

Lors de la **configuration Online,** la connexion reste établie pendant la configuration du bus et le paramétrage des participants.

Dans l'exemple suivant la configuration est réalisée offline.

9.1 Installation d'un fichier EDS

En raison d'une erreur dans le DeviceNet-Manager 2 fichiers EDS se trouvent sur la disquette :

- 1.EDS: Fichier EDS standard selon la spécification (Utiliser ce fichier pour des programmes tels que RS-Networx ou autres
- DNetMan.EDS: utiliser ce fichier EDS uniquement pour le DeviceNet-Manager

Pour installer le fichier EDS choisir dans le menu principal *Utilities* la rubrique *Install EDS Files...* Dans la fenêtre suivante cliquer le fichier de la disquette **DNetMan.eds**.

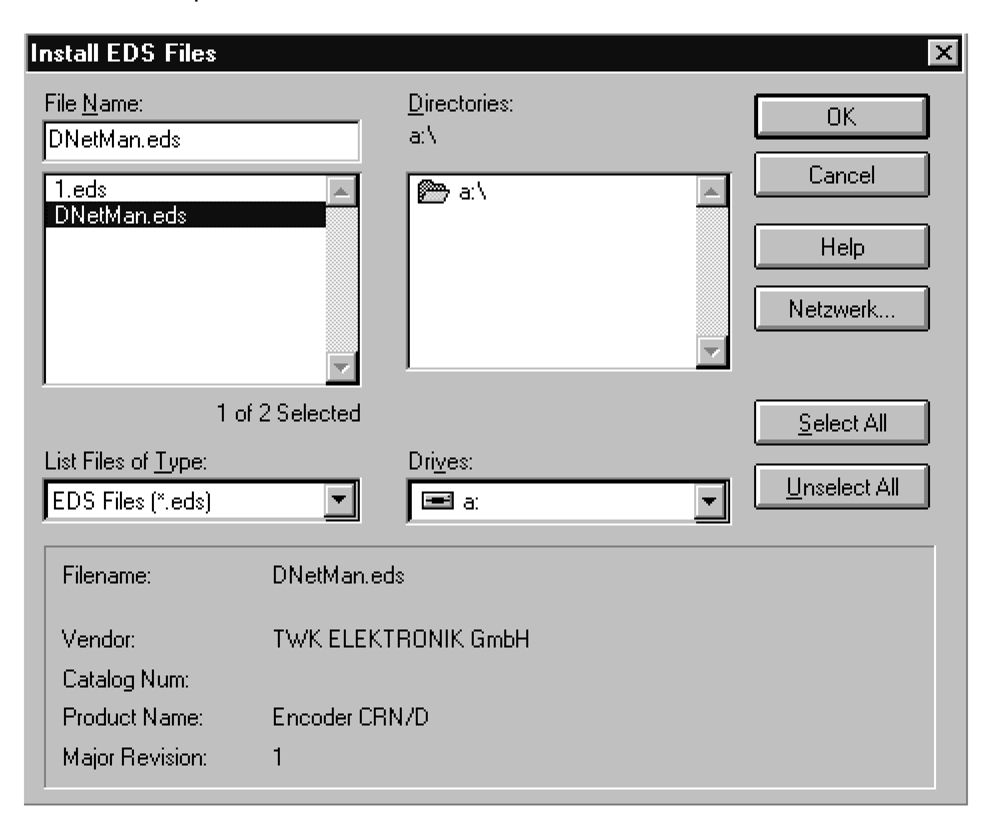


Illustration 9.1 Installation du fichier EDS

Répondre „oui“ à la question d'attribuer ou non un Bitmap au nouvel appareil. La fenêtre suivante montre le dialogue dans lequel il est possible de cliquer le Bitmap, présent lui aussi sur la disquette, **crn_d_m** (modèle de codeur avec connecteur) ou le Bitmap **crn_d_z** (modèle de codeur avec boîtier de raccordement).



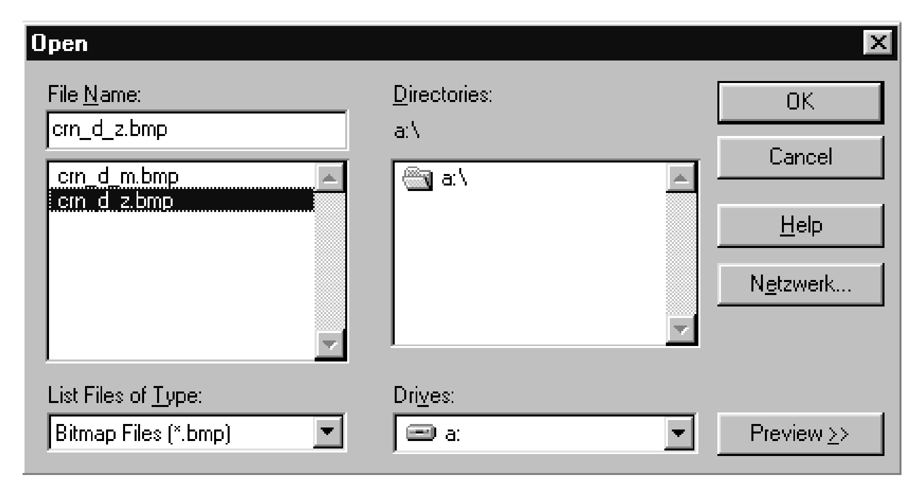


Illustration. 9.2 Installation du symbole du codeur

L'installation du fichier EDS et du symbole du participant est maintenant terminée.

9.2 Raccordement au bus

Après avoir installé un projet, il est possible de choisir dans la *Device List* sous *Generic*, *TWK Elektronik GmbH* le codeur *Encoder CRN/D* et de le raccorder au bus via Drag & Drop.

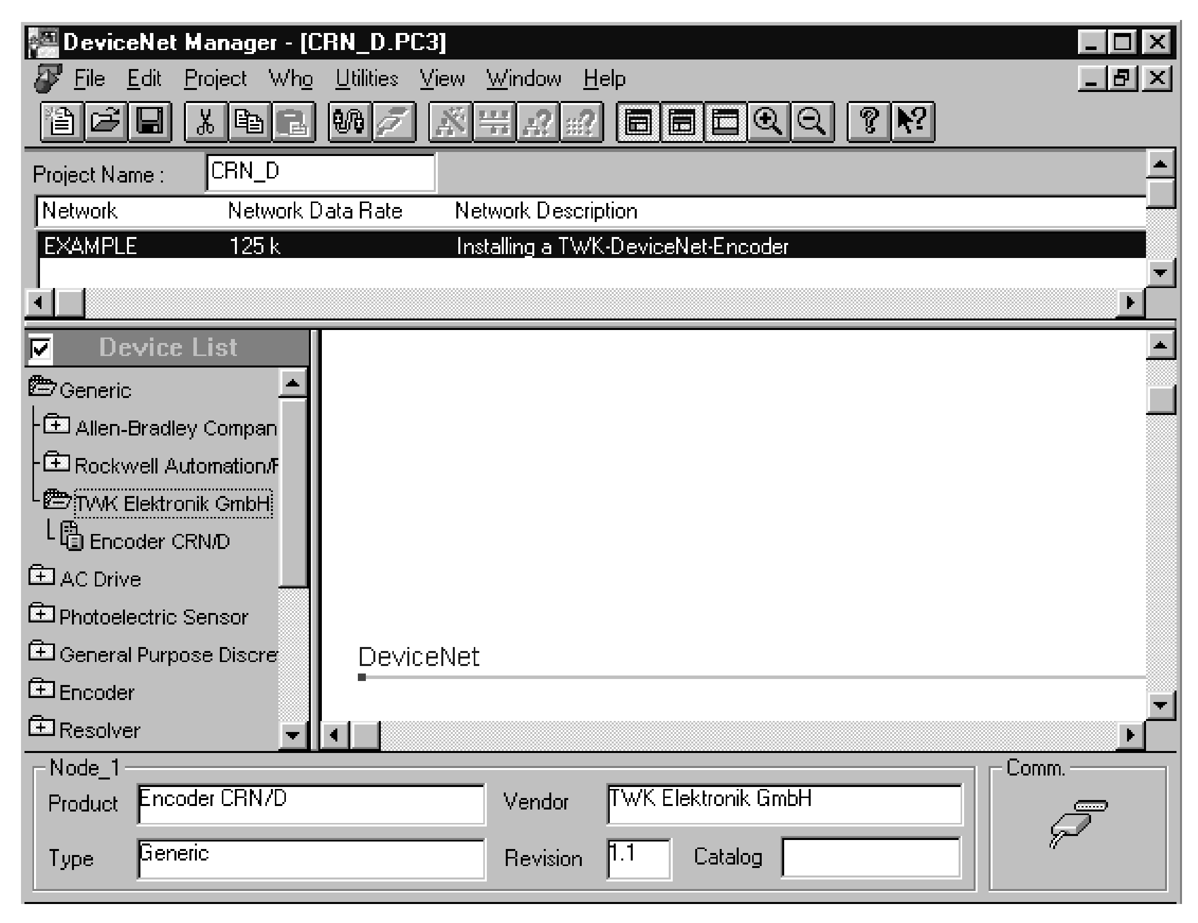


Illustration 9.3 Raccordement du codeur

Il faut ensuite entrer le nom et l'adresse du participant. Comme adresse entrer l'adresse donnée au codeur (via commutateur Dip dans le boîtier, via Software pour le modèle de codeur avec connecteur (voir chapitre 9.5)).



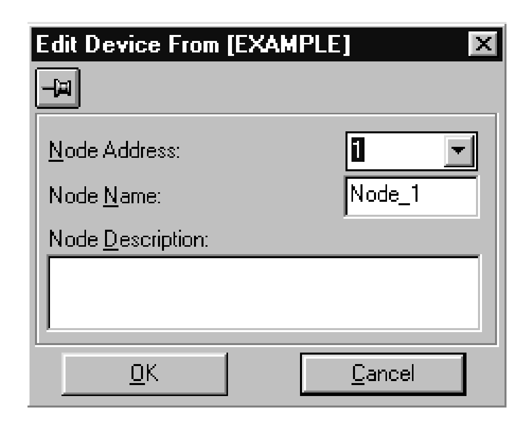


Illustration 9.4 Adressage

Dans ce cas de figure l'adresse par défaut 1 du codeur a été conservée et le nom proposé comme Node Name, „Node_1", accepté. Après avoir confirmé avec OK le codeur apparaît dans le bus.

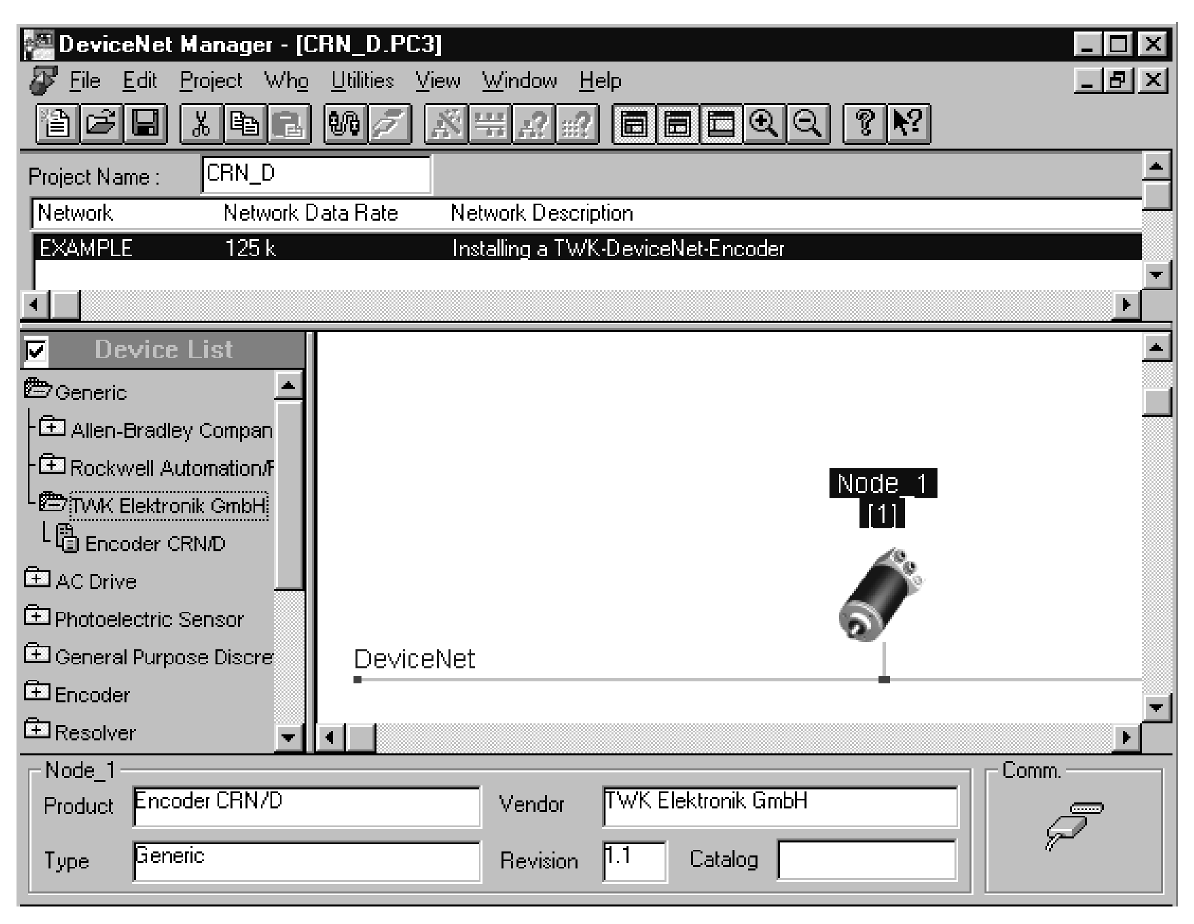


Illustration 9.5 Codeur dans le bus DeviceNet

9.3 Paramétrage du codeur

Dans le DeviceNet-Manager, 2 possibilités pour modifier les paramètres d'un participant dans DeviceNet sont à la disposition de l'utilisateur :

- **Basic Device Configuration**
- **Enhanced Device Configuration**

La **Basic Device Configuration** n'est possible qu'en mode online. Dans ce cas de figure il est nécessaire de savoir où se trouvent les paramètres (attributs) dans le modèle d'objet de DeviceNet, et aussi de connaître la fonction (service) qui permet de lire ces paramètres.



L'**Enhance Device Configuration** est une possibilité confortable de modifier et de lire les paramètres du codeur et ceci aussi bien online qu'offline (seulement en fichier). Dans ce mode de configuration les paramètres peuvent être enregistrés dans un fichier (*.dcf) pour être transmis plus tard au participant.

Une commande **Save** permet de mémoriser les paramètres dans l'EEPROM du codeur pour parer à une coupure de secteur. Par ailleurs il est toujours possible de charger à nouveau les valeurs par défaut de tous les paramètres grâce à la commande **Restore**.

Ces deux commandes ne peuvent être utilisées qu'avec le Basic Device Configurator. (voir chapitres 9.5 et 9.6)

Ouvrir l'Enhanced Configurator en cliquant deux fois sur le symbole du codeur. En mode offline les paramètres sont pourvus à l'avance des valeurs par défaut issues du fichier EDS. La fenêtre suivante apparaît alors :

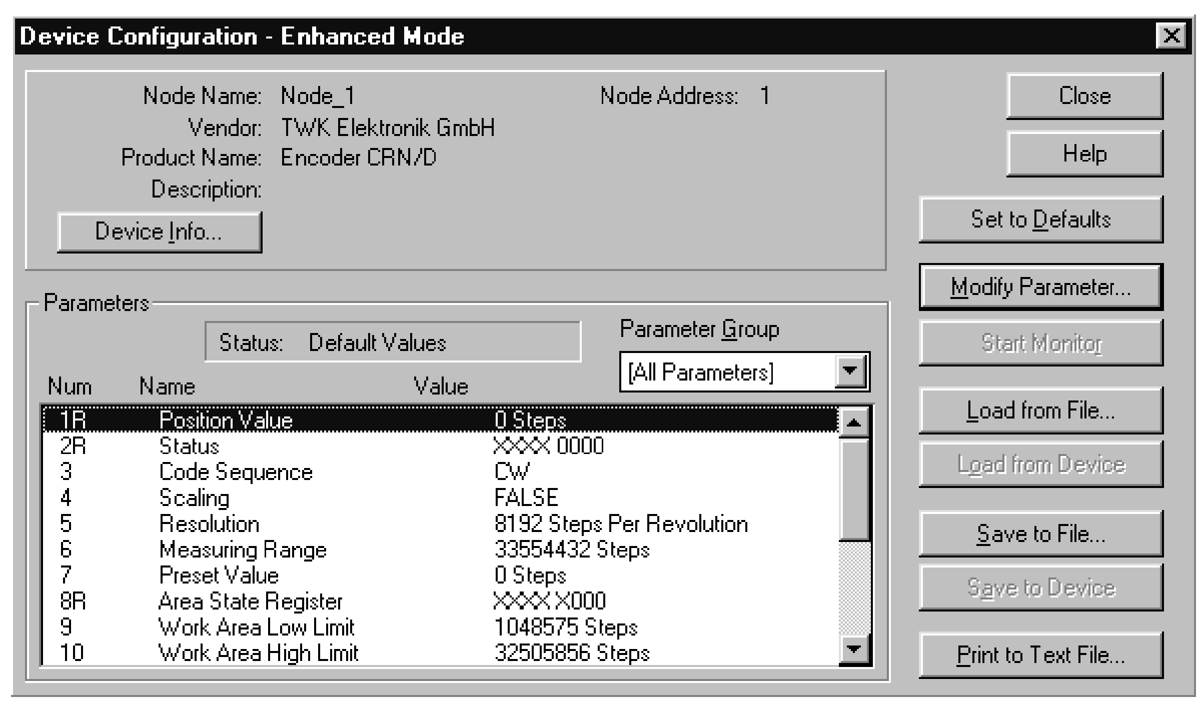


Illustration 9.6 Paramètres du codeur

Les valeurs caractérisées par un „R“ représentent les informations d'état du codeur, elles ne peuvent qu'être lues. Toutes les autres valeurs sont des paramètres modifiables. En cliquant deux fois sur un paramètre ou un activant la touche *Modify Parameter*, il est possible de modifier leur valeur. Selon qu'il s'agit d'un paramètre de type numérique ou Bool apparaissent les fenêtres suivantes :

Paramètre de type Bool :

Pour modifier la résolution ou la résolution totale du codeur, activer „ON“ dans le paramètre #4 „Scaling“.

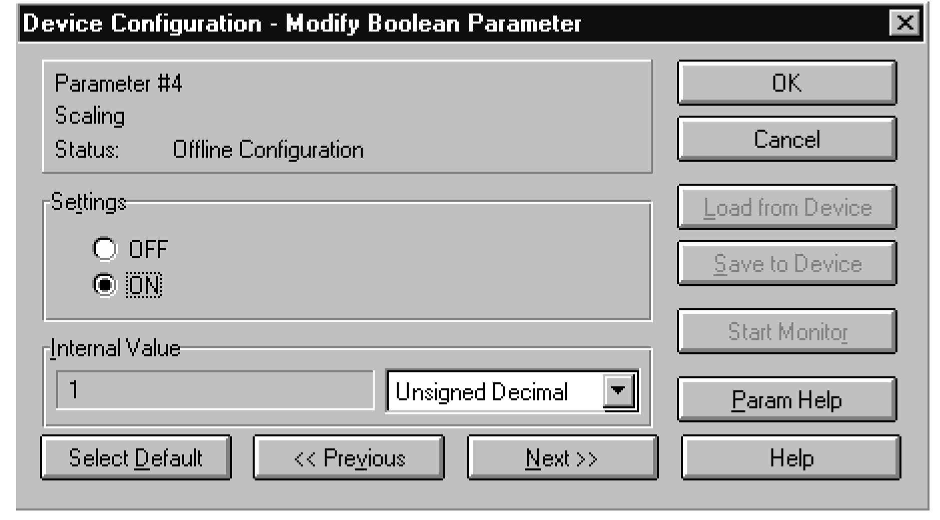


Illustration 9.7 Modification d'un paramètre de type Bool



Paramètre de type numérique :

La modification de la valeur du paramètre de type numérique est réalisée en entrant un nombre ou en déplaçant le curseur. Pour cet exemple la résolution a été fixée à 4096 pas/tour.

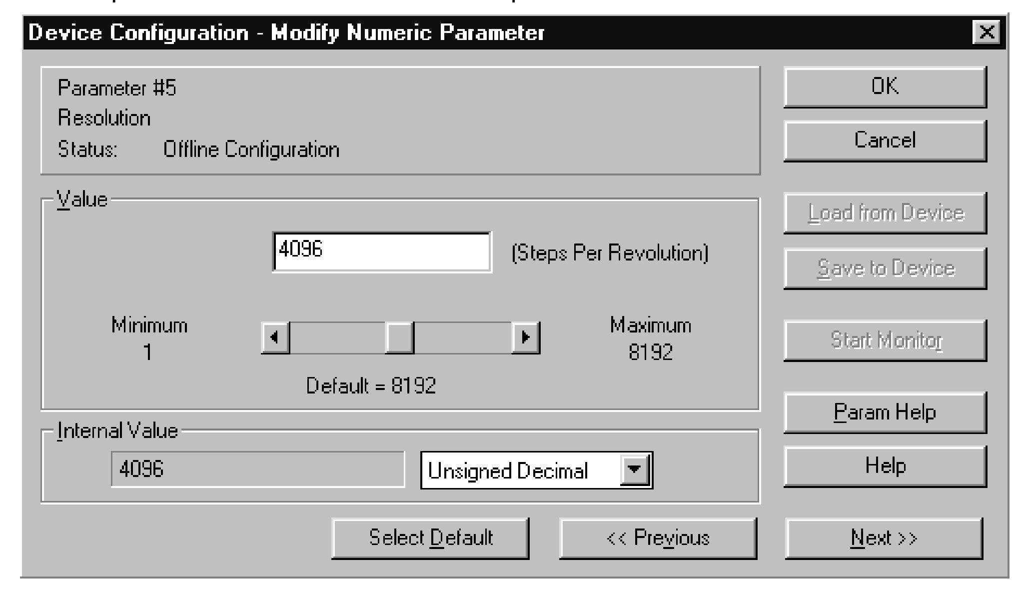


Illustration 9.8 Modification d'un paramètre de type numérique

Après avoir configuré tous les paramètres il est possible de les enregistrer dans un fichier via la touche *Save to File...* de l'Enhanced Configurator. Chaque participant a son propre fichier (dans ce cas de figure Node_1.dcf).

9.4 Introduction du codeur dans la Scanlist

Pour définir le mode de fonctionnement du codeur et l'assigner à un Master, il faut introduire le codeur dans la Scanlist d'un Master.

La fenêtre ci-dessous montre la structure du bus avec un automate SLC500 de chez Allen-Bradley et un Master 1747-SDN Scanner Modul. Toutes les fenêtres qui suivent se réfèrent à cette configuration.

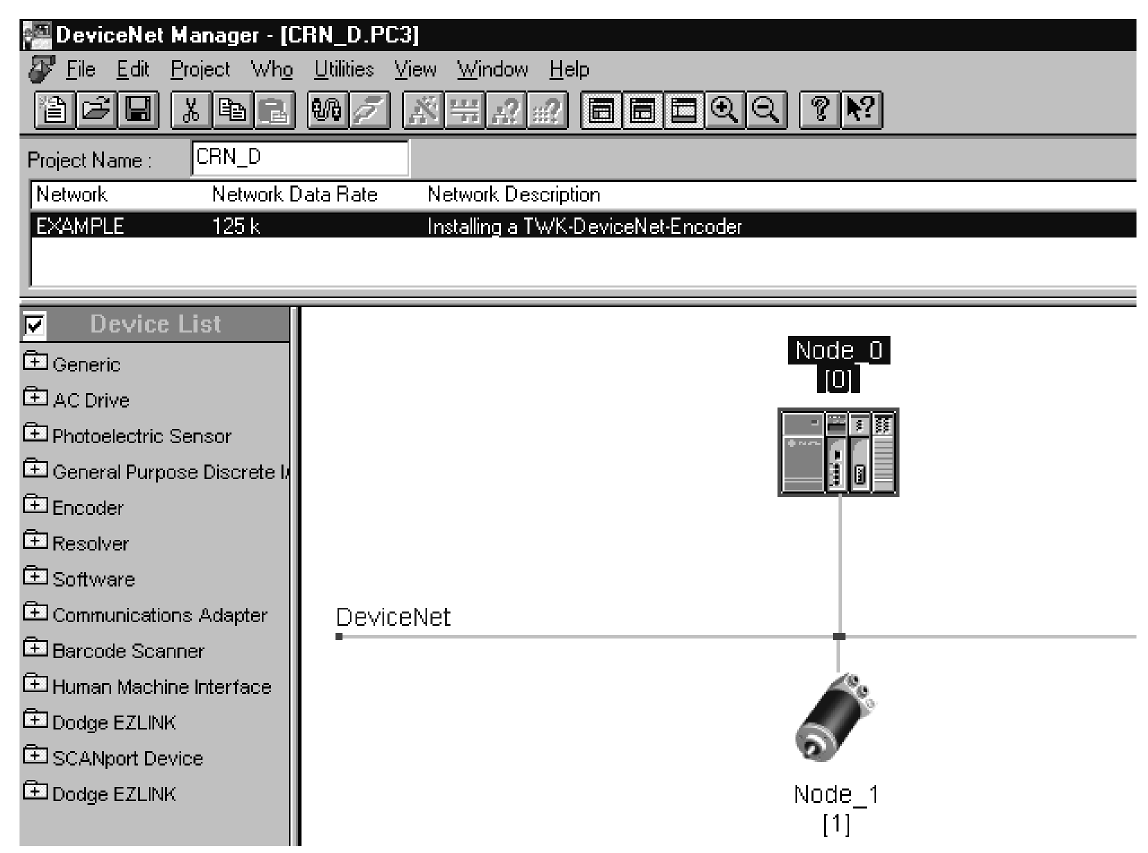


Illustration 9.9 Structure du bus avec Master



Cliquer deux fois sur le symbole de l'automate pour faire apparaître le dialogue ci-dessous qui configure les paramètres du Master. A titre d'exemple les configurations ci-dessous peuvent être reprises et enregistrées dans un fichier (ici Node_0.sm4).

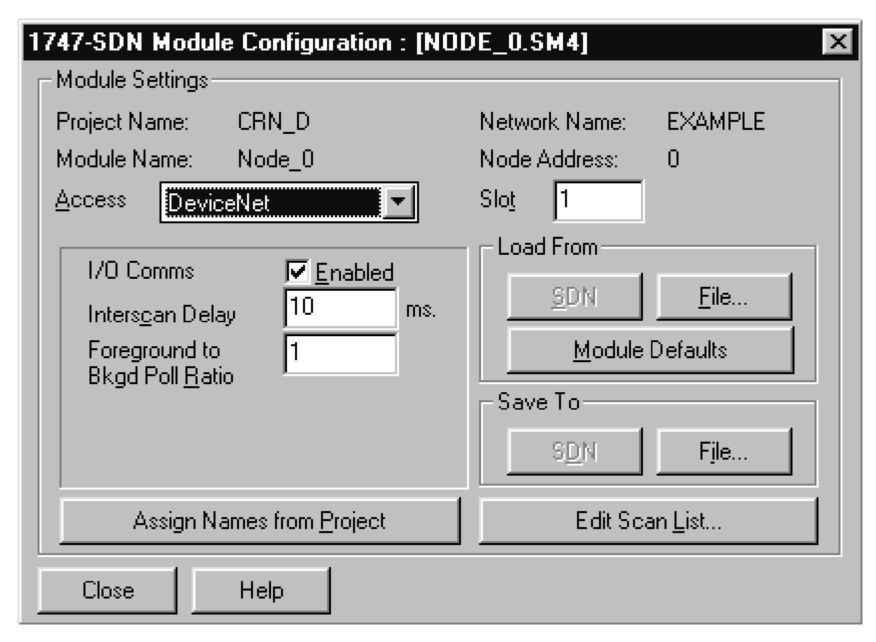


Illustration 9.10 Paramétrage du Master

Activer la touche *Edit Scan List...* pour faire apparaître le **Scan List Editor**. Puisqu'aucun participant n'a été introduit dans cette liste la fenêtre est vide.

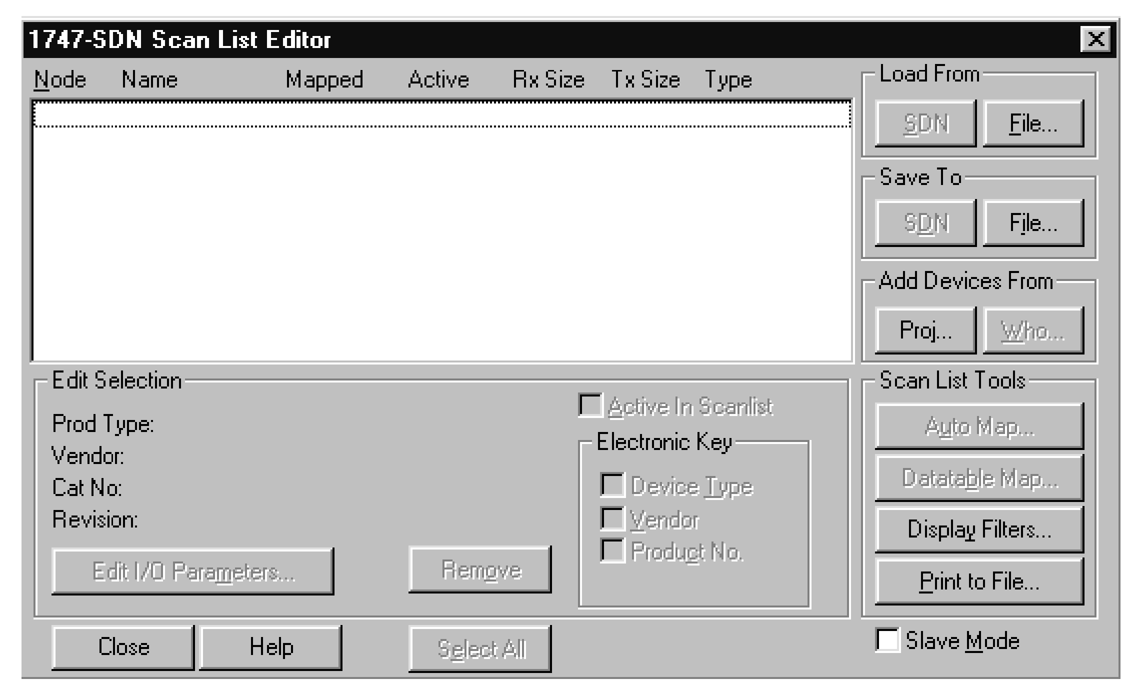


Illustration 9.11 Scanlist du Master



Pour introduire un participant à la Scan List, activer la touche *Proj...* du groupe de touches *Add Devices From*. Le schéma de la structure du bus apparaît alors à nouveau :

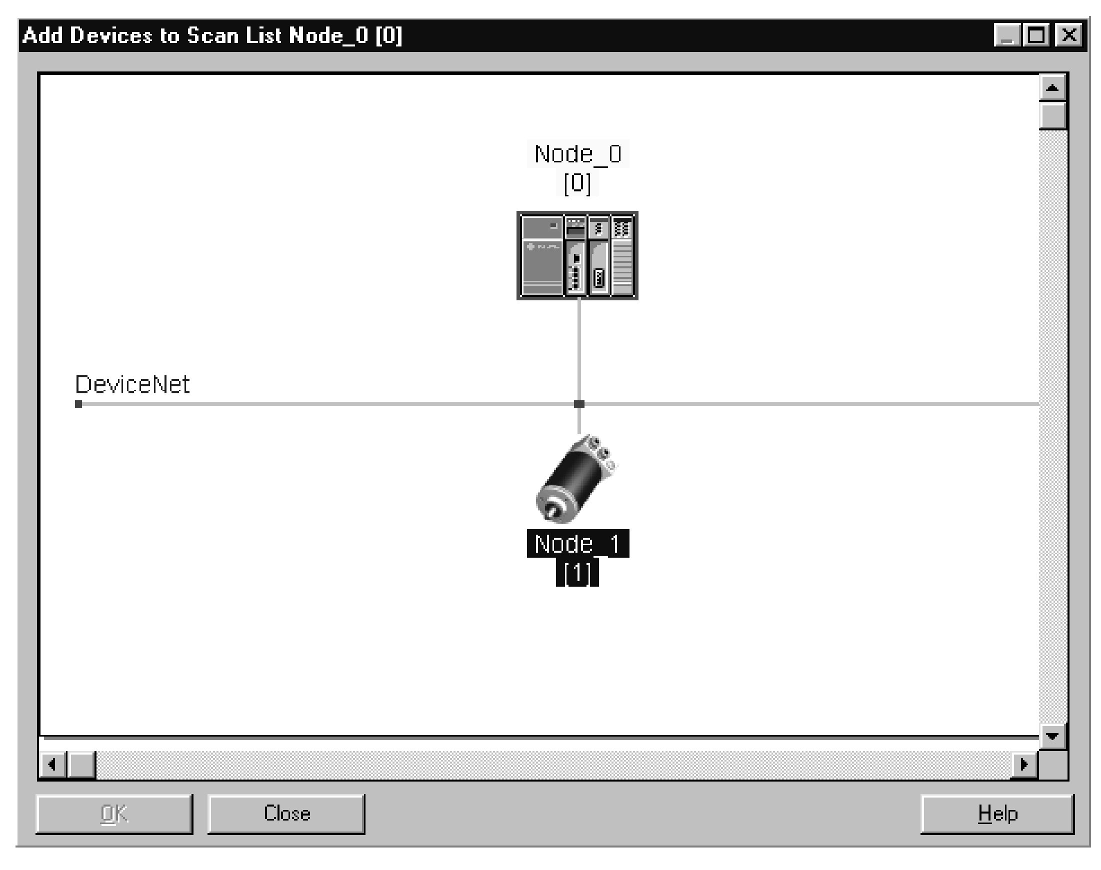


Illustration 9.12 Introduire le codeur dans la Scanlist

Glisser le codeur via Drag & Drop sur le symbole du Master. Le codeur reçoit alors un cadre rouge et le nombre 0. Ceci montre que le participant de la Scanlist du Master a été assigné de l'ID 0. Quitter ensuite cette fenêtre en cliquant sur OK.

A présent le codeur apparaît dans le Scan List Editor avec son mode de fonctionnement par défaut, le Poll-Mode.

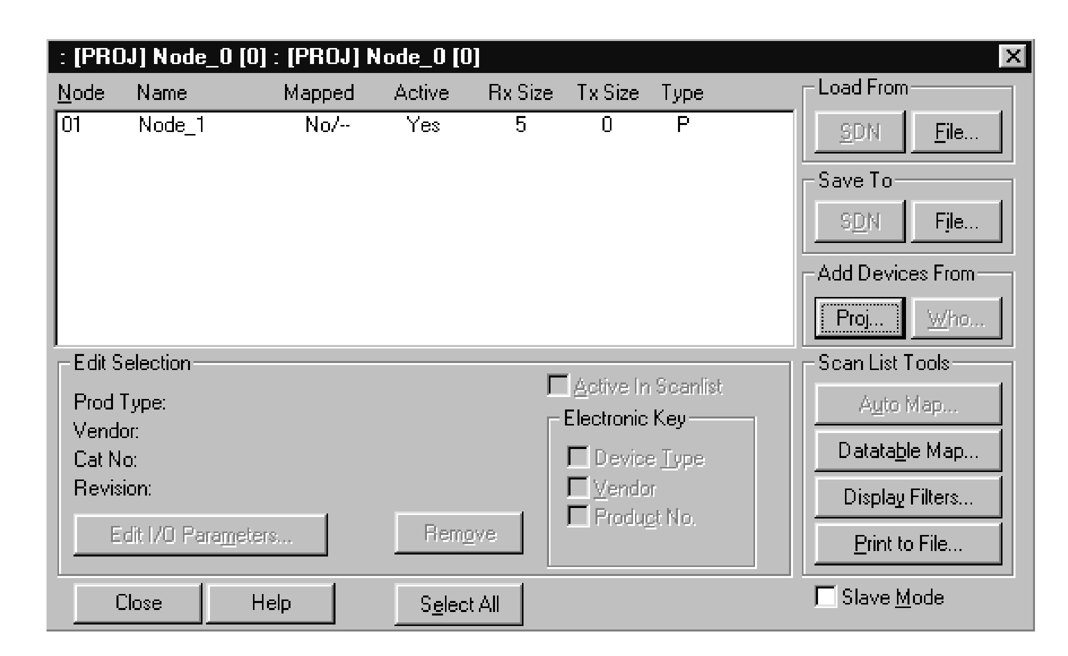


Illustration 9.13 Scanlist avec codeur



Pour modifier le mode de fonctionnement du codeur cliquer deux fois sur la 1ère ligne. Dans la fenêtre suivante il est possible de choisir entre les différents modes de fonctionnement Poll-Mode, Bit-Strobe-Mode et Change of State-Mode/ Cyclic Mode. Il est également possible de choisir plusieurs modes de fonctionnement simultanément. Le nombre des données d'entrée (Rx) doit toujours être 5 octets et celui des données de sortie 0 octet.

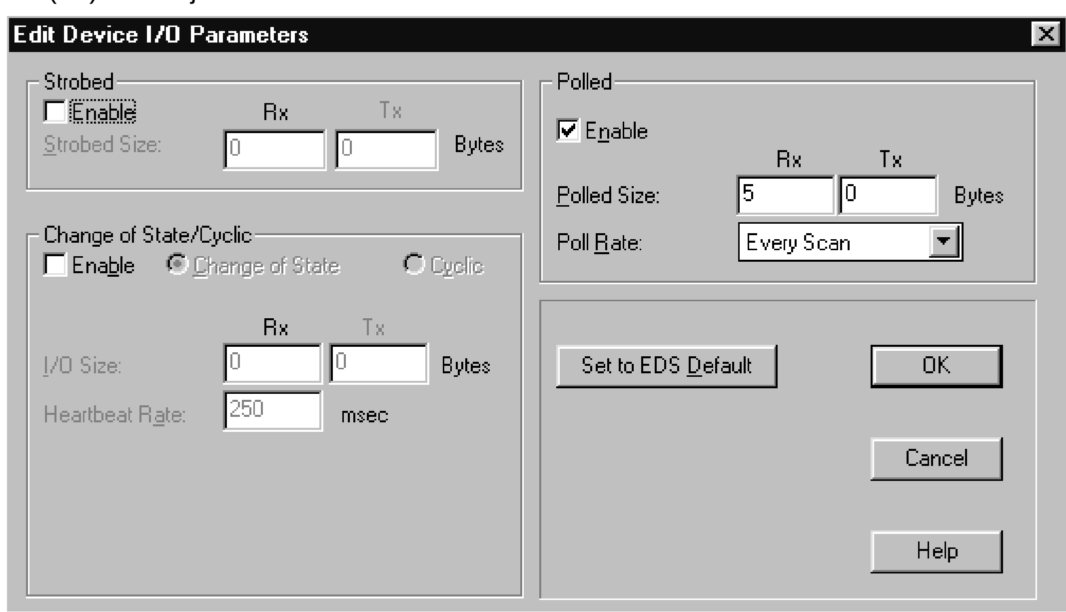


Illustration 9.14 Fixer le mode de fonctionnement du codeur

Le codeur est maintenant dans la Scan List du Master affecté du mode de fonctionnement souhaité. Afin d'avoir accès aux données de l'automate, il faut indiquer au Scanner où se trouvent les données du codeur dans l'entrée de l'automate. Pour se faire existe le **Datatable Map** qui peut être réalisé manuellement ou automatiquement. Pour cet exemple, le procédé automatique suffit, accepter pour cela les données standards.

Enregistrer les configurations effectuées dans un fichier (ici Node_0.sl4).

9.5 Aller online et transmettre les données

Après avoir enregistré dans un fichier les paramètres du codeur, les réglages du Master et la Scanlist, ce fichier est transmis au participant de la façon suivante :

Transmission de la Scanlist au Master :

Entrer en ligne avec le réseau, aller dans le Scan List Editor et charger la Scanlist réalisée dans le chapitre 9.4. La fenêtre suivante apparaît :

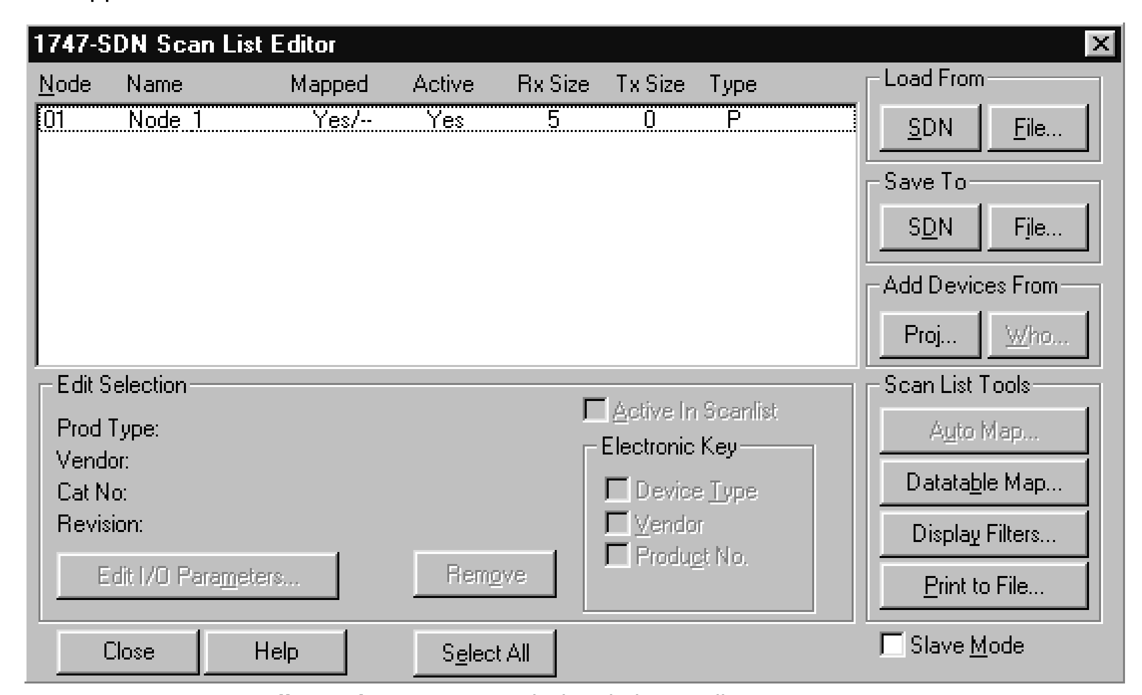


Illustration 9.15 Transmission de la Scanlist au Master

Transférer la Scan List au Scanner (Master) avec la commande *Save to SDN*. Pour un nouveau projet entrer *All Records* dans la fenêtre suivante.

Transmettre les paramètres au codeur :

Cliquer deux fois sur le symbole du codeur pour ouvrir le Device Configurator - Enhanced Mode. Puisque la connexion est établie, il est possible de lire les paramètres actuels du codeur .

Charger les valeurs des paramètres enregistrées dans le fichier Node_1.dcf (voir chapitre 9.3). La fenêtre ci-dessous apparaît :

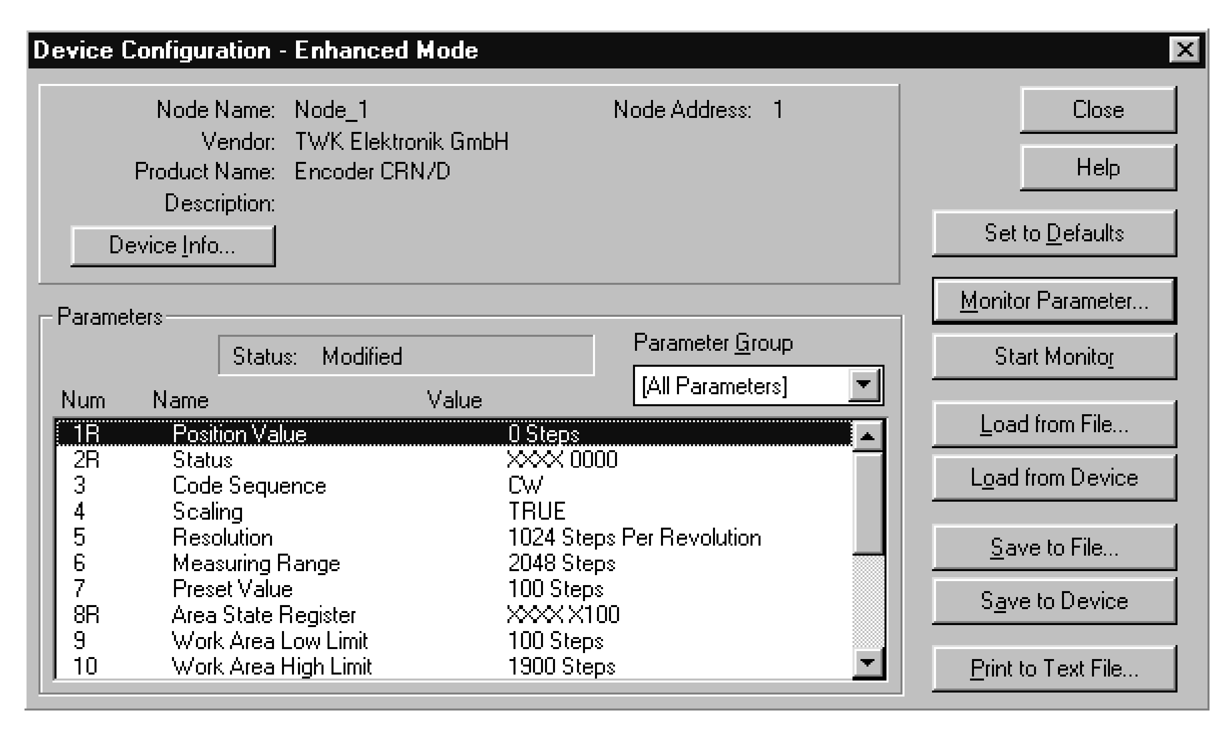


Illustration 9.16 Transmission des paramètres au codeur

Transmettre ensuite les paramètres au codeur grâce à la touche *Save to Device*. Les nouvelles valeurs sont transférées dans la RAM du codeur et sont immédiatement valables. Afin qu'elles soient également actives en cas de coupure de secteur, il est possible de les enregistrer dans l'EEPROM du codeur grâce à une commande spéciale (voir chapitre 9.7).

9.6 Adressage et réglage de la vitesse de transmission d'un codeur avec connecteur

Le modèle CRN/D avec connecteur permet l'adressage (MAC-ID) et le réglage de la vitesse de transmission (125 kB, 250KB, 500KB) via Software.

Le réglage de la vitesse de transmission devrait toujours avoir lieu lors d'une connexion point par point avec le participant car la modification de la vitesse de transmission d'un participant du réseau peut anéantir tout le fonctionnement du bus. L'adressage d'un participant peut être, lui, effectué pendant le fonctionnement du bus. Faire attention cependant de ne pas utiliser une adresse déjà attribuée à un autre participant.

La modification de la vitesse de transmission n'est active qu'après une nouvelle mise sous tension on/off, la modification de l'adresse prend effet immédiatement.

Etablir une connexion point à point avec le codeur et aller Online avec le DeviceNet Manager. Fermer tout éventuel projet ouvert et cliquer *sur Node Commissioning* dans le menu *Utilities*.



Modification de l’adresse du participant :

Entrer dans le champ gauche *Node Address* l’adresse actuelle du codeur (l’adresse par défaut est 1) et dans le champ droit l’adresse souhaitée. La nouvelle adresse du codeur est valide après confirmation grâce à la touche *Apply Node Settings* .

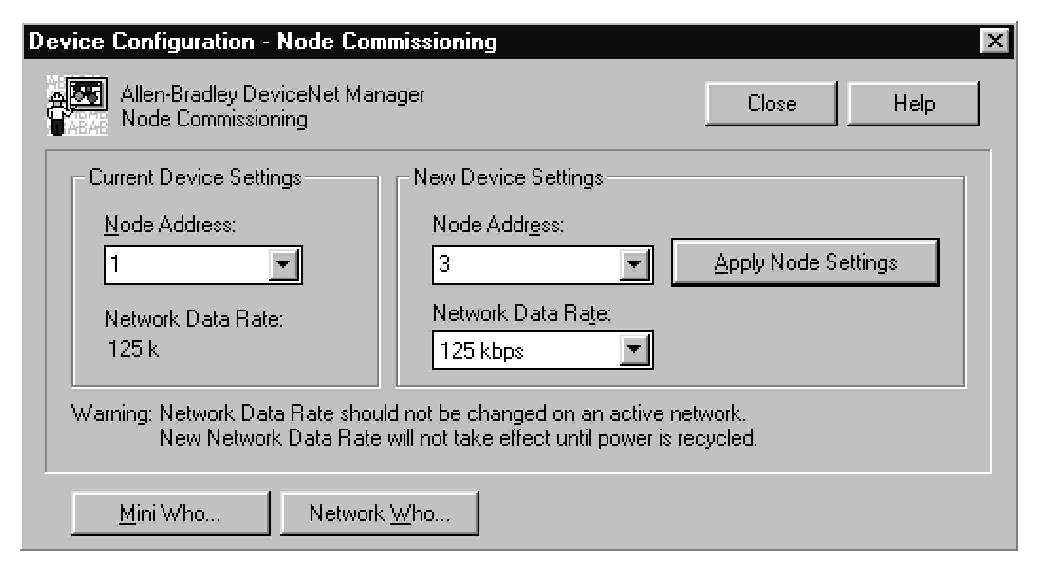


Illustration 9.17 Adressage

Modification de la vitesse de transmission :

L’actuelle vitesse de transmission apparaît dans le champ *Current Device Settings* (ici 125 KB). Entrer sous *New Device Settings* la vitesse de transmission souhaitée. Après confirmation grâce à la touche *Apply Node Settings* apparaît à nouveau la mise en garde de modifier cette configuration seulement lors d’une connexion point par point. Après confirmation la nouvelle vitesse de transmission de codeur est enregistrée mais active seulement après la remise sous tension off/ on.

Ces deux modifications sont réalisables également de façon simultanée. Un enregistrement supplémentaire de l’adresse du participant et de la vitesse de transmission n’est pas nécessaire.

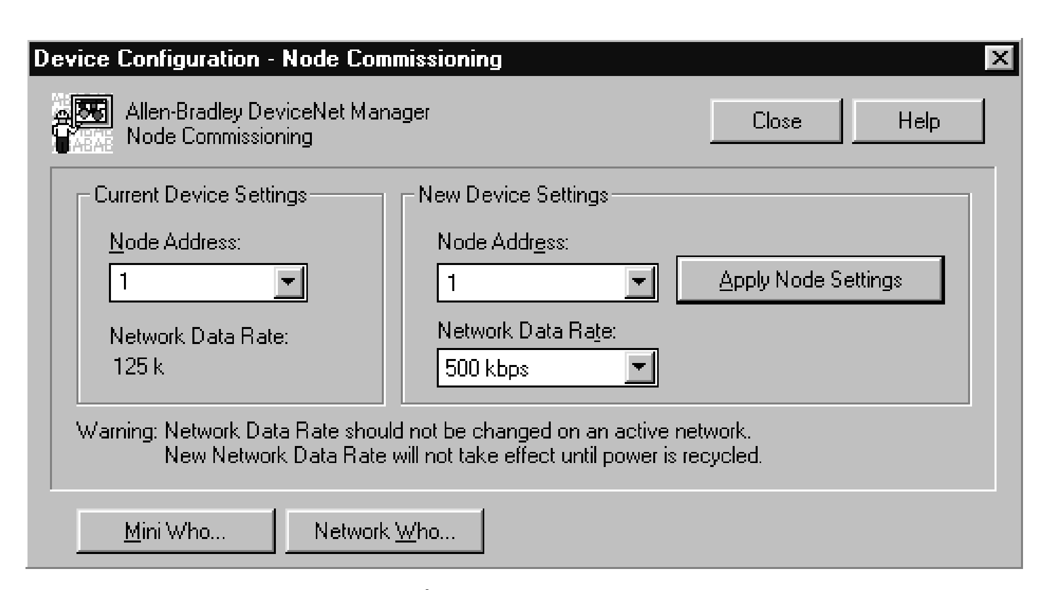


Illustration 9.18 Réglage de la vitesse de transmission



9.7 Enregistrer les paramètres dans l'EEPROM

Cet enregistrement est effectué via le Device Configurator - Basic Mode.

Online, aller sous le menu principal *Utilities* dans la rubrique *Basic Device Configuration*. Entrer l'adresse du codeur, la Class 35_{dez} (Position Sensor Object) et le service 22_{dez}. Activer ensuite la touche *Save to Device*. (voir illustration)

L'enregistrement des paramètres est indiqué dans la ligne d'état du DeviceNet Manager.

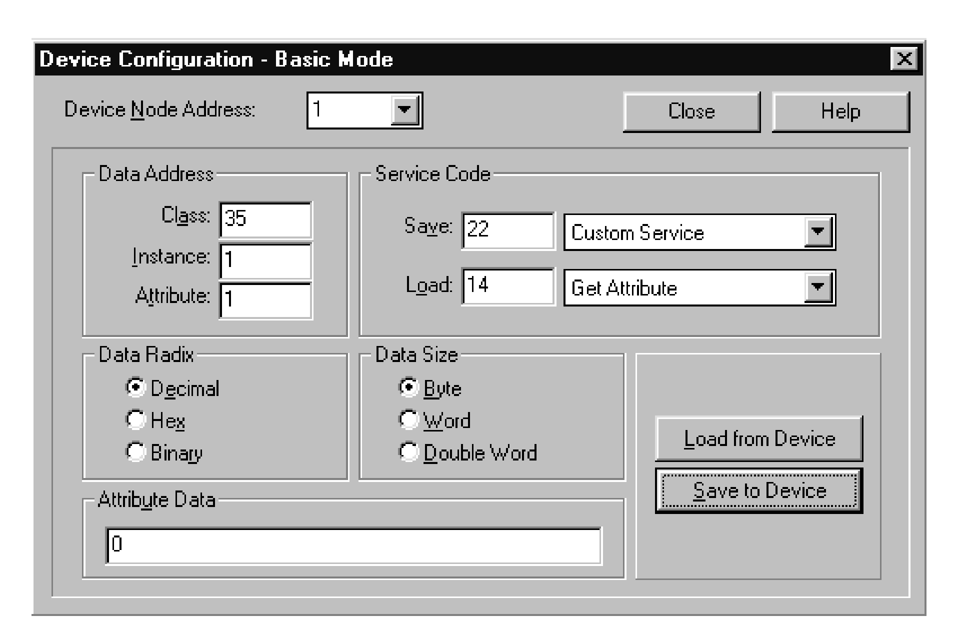


Illustration 9.19 Enregistrement des paramètres dans l'EEPROM

9.8 Charger les valeurs par défaut des paramètres

Il est possible de réinstaller toutes les valeurs par défaut des paramètres du codeur avec la commande „Restore“ Service Code 21_{dez} du Position-Sensor-Object.

Online, aller sous le menu principal *Utilities* dans la rubrique *Basic Device Configuration*. Entrer l'adresse du codeur et le Service 21_{dez} de la Class 35_{dez} (Position Sensor Object). Activer ensuite la touche ***Save to Device***.

Les valeurs par défaut sont actives après avoir éteint et remis en marche le codeur (tension off/on).

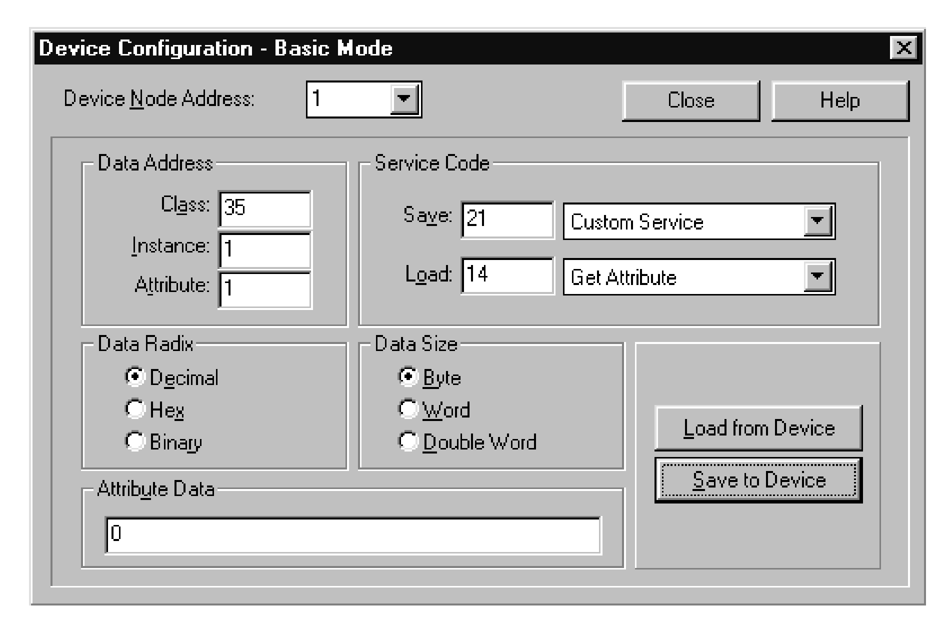


Illustration 9.20 Chargement des valeurs par défaut des paramètres



9.9 Lire un attribut

Via le Basic Configurator il est possible de lire tous les attributs (caractérisés comme lisibles) des divers DeviceNet-Objects présents dans le codeur. La liste des objets et des Services attenants se trouve décrite dans le chapitre 8.

La lecture de tous les attributs s'effectue grâce au service **Get Attribute** (Service-ID 14_{dez}), l'ecriture grâce au service **Set Attribute** (Service-ID 16_{dez}).

Dans l'exemple ci-dessous il s'agit de lire le numéro de série (Attribut 6 de l'Identity Object).

Online, aller sous le menu principal *Utilities* dans la rubrique *Basic Device Configuration...* . Entrer l'adresse du codeur, Class 1 (Identity Object), Instance 1 et Attribute 6. Changer la représentation (Data Radix) en Hex et la taille des données (Data Size) en Double Word. Le Service 16_{dez}, à utiliser, est la configuration par défaut.

Activer ensuite la touche *Load from Device*.

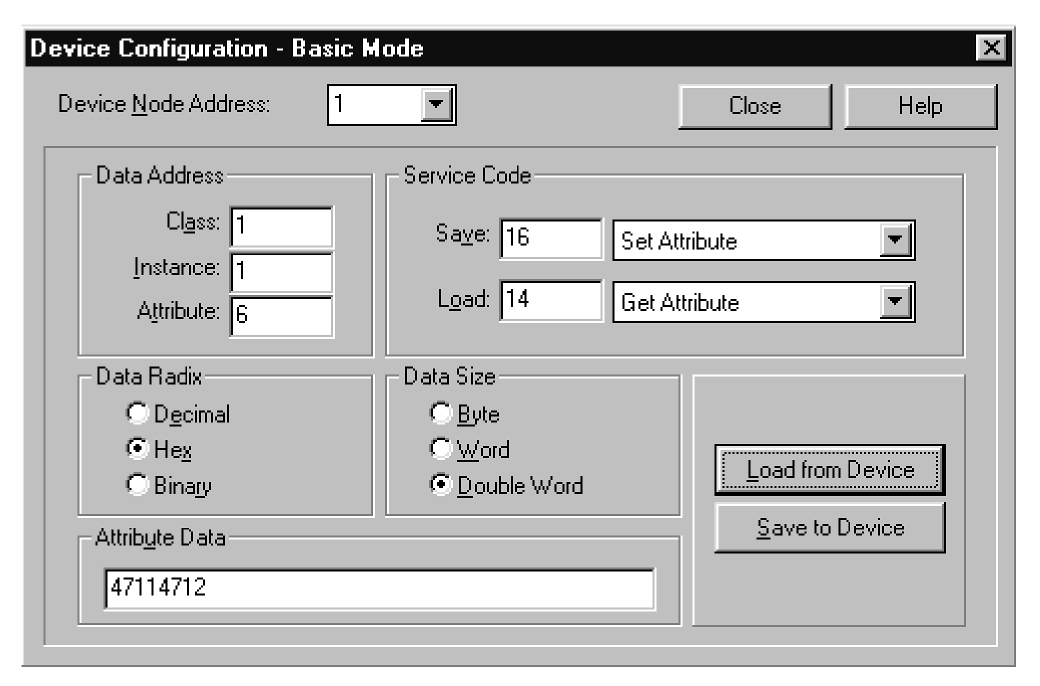


Illustration 9.21 Lire le numéro de série



10. RS-Networx for DeviceNet

Ce chapitre décrit le raccordement du codeur TWK CRN/D au système DeviceNet-Master-Slave d'un automate d'Allen-Bradley grâce au programme RS-Networx for DeviceNet. La connaissance exacte des objets, thème du chapitre 8, n'est pas nécessaire.

10.1 Installation du fichier EDS

En raison d'une erreur dans le DeviceNet-Manager 2 fichiers EDS se trouvent sur la disquette :

- 1.EDS: Fichier EDS standard selon la spécification (Utiliser ce fichier pour des programmes tels que RS-Networx ou autres)
- DNetMan.EDS: utiliser ce fichier EDS uniquement pour le DeviceNet-Manager

Pour installer le fichier EDS choisir dans le menu principal *Tools* puis la rubrique *EDS Wizard... .* Plusieurs fenêtres guident l'utilisateur dans l'installation du fichier EDS et du symbole du codeur. Dans le dialogue illustré ci-dessous entrer comme nom du fichier **A:\1.EDS**.

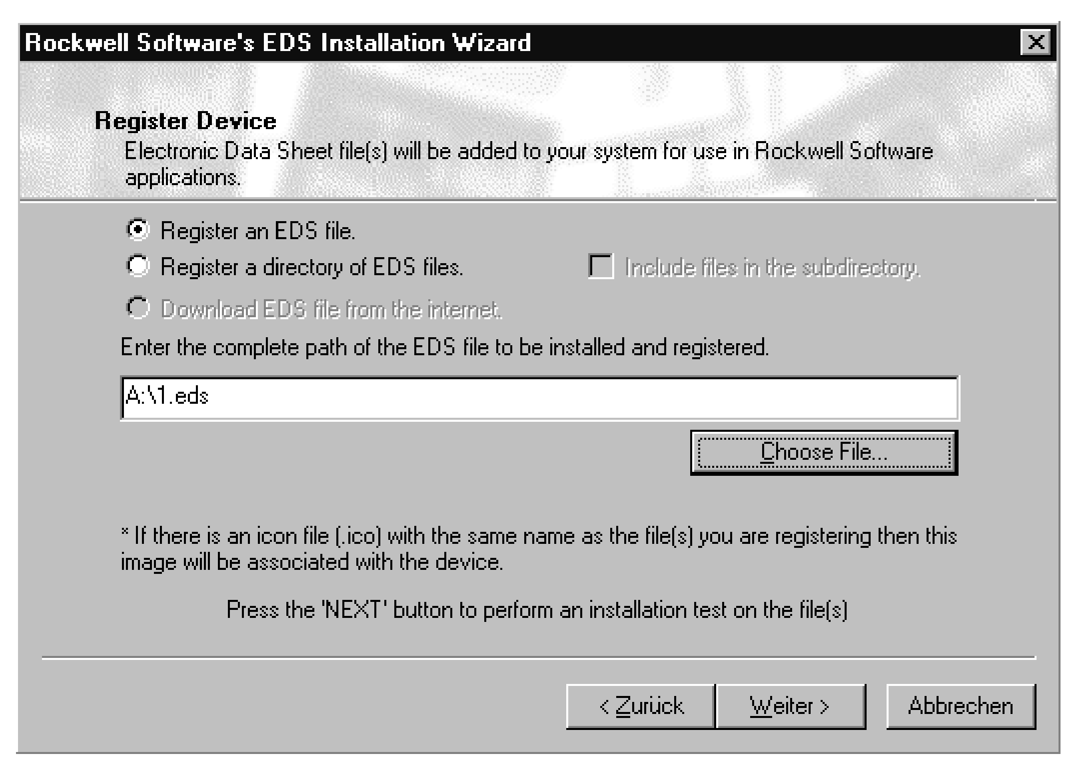


Illustration 10.1 Installation du fichier EDS

Choisir ensuite entre les 2 codeurs dont les symboles sont présents sur la disquette :

- crn_d_m.ico pour le modèle de codeur avec connecteur
- crn_d_z.ico pour le modèle de codeur avec boîtier de raccordement

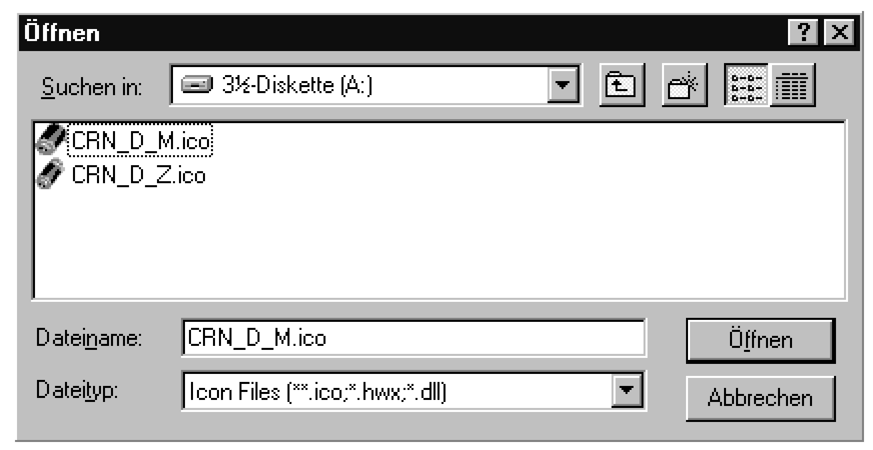


Illustration 10.2 Installation du symbole du codeur



10.2 Raccordement au bus

Après l'installation du fichier EDS le codeur apparaît dans le catalogue du Hardware sous *Generic Device / Encoder CRN/D.* Il peut alors être raccordé au bus via Drag & Drop.

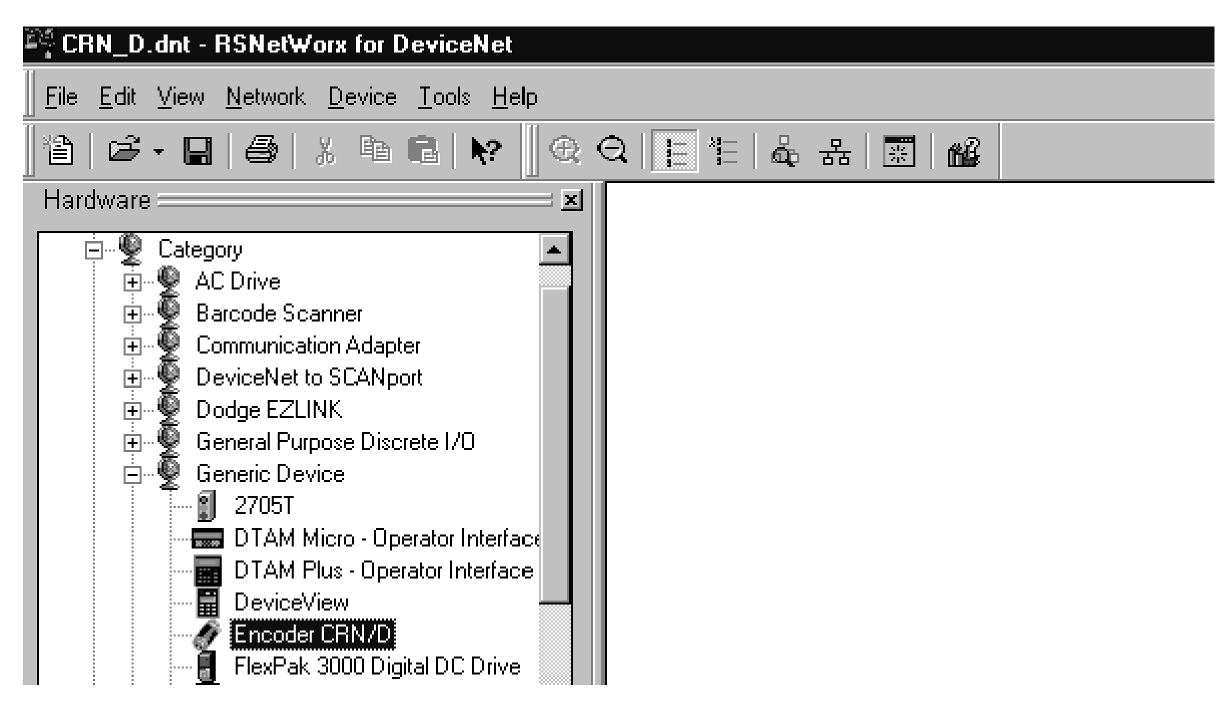


Illustration 10.3 Raccordement du codeur

Lors du raccordement du codeur celui-ci obtient automatiquement la prochaine adresse libre. Pour changer l'adresse et paramétrer le codeur cliquer deux fois sur le symbole du codeur présent dans le bus, la fenêtre illustrée ci-dessous apparaît alors. Aller dans *General / Address* et entrer l'adresse préréglée du codeur. Dans cet exemple, l'adresse utilisée est l'adresse par défaut 1.

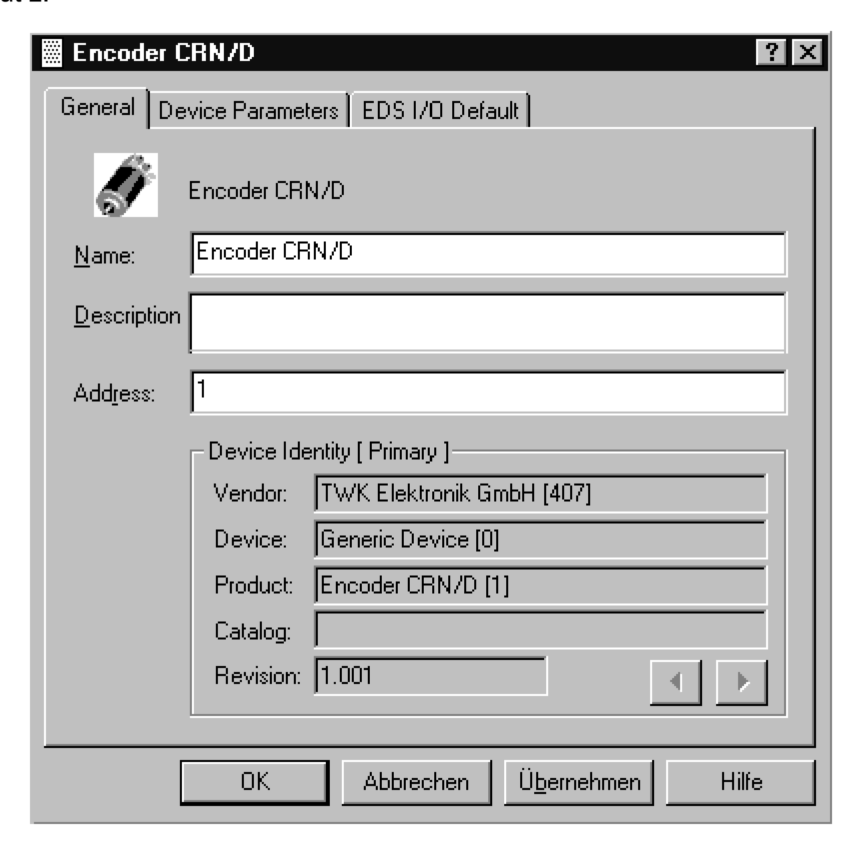


Illustration 10.4 Adressage

10.3 Paramétrage du codeur

Cliquer sur le registre *Device Parameters* (voir illustration 10.4) pour obtenir la fenêtre suivante :

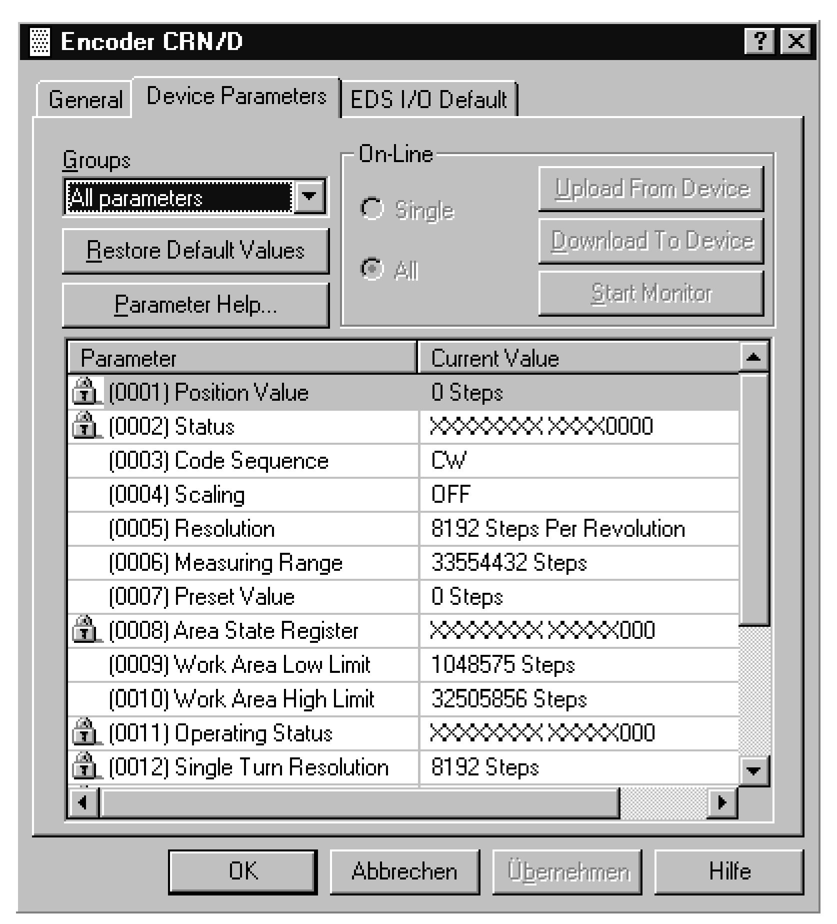


Illustration 10.5 Paramètres du codeur

Dans cette fenêtre sont indiqués tous les paramètres et les informations d'état du codeur (dans cet exemple il s'agit des paramètres d'un codeur 25-Bit). Dans le champ *Groups* il est possible de réduire l'affichage aux paramètres seuls ou aux informations d'état seules.

Après le raccordement du codeur au bus et sans connexion online avec le bus sont indiquées ici les valeurs par défaut des paramètres et des informations d'état telles qu'elles se trouvent dans le fichier EDS. En cliquant deux fois sur un paramètre, il est possible de modifier sa valeur. Les informations d'état ne peuvent qu'être lues et sont caractérisées par un cadenas. Grâce à la touche *Restore Default Values* il est à tout moment possible de lire à nouveau les valeurs par défaut du fichier EDS.

Grâce à la touche *Parameter Help* il est possible de faire apparaître un texte d'aide concernant chaque paramètre.

Lors d'une connexion online avec le bus il est possible de lire les paramètres et les informations d'état du codeur (*Upload From Device*) ou bien de transmettre les paramètres au codeur (*Download To Device*). Dans ce cas une actualisation cyclique des valeurs est également réalisable grâce à la touche *Start Monitor*.

Une commande **Save** permet de mémoriser les paramètres dans l'EEPROM du codeur pour parer à une coupure de secteur. Par ailleurs il est toujours possible de charger à nouveau les valeurs par défaut de tous les paramètres grâce à la commande **Restore**. Ces commandes ne sont utilisables qu'avec le „Class Instance Editor" (voir chapitres 10.7 et 10.8).

Le registre *EDS I/O Default* indique uniquement le nombre d'octets que le codeur envoie ou reçoit selon les divers modes de fonctionnement. La configuration du mode de fonctionnement n'est possible que dans la Scanlist du Master (voir chapitre 10.4).

Modifier les paramètres selon les exigeances de l'utilisateur et quitter la fenêtre avec OK. Il n'est pas nécessaire de transmettre séparément les paramètres au codeur. Dans RS-Networx une seule commande permet de transmettre tous le projet du bus (voir chapitre 10.5) .



10.4 Introduction du codeur dans la Scanlist

Pour définir le mode de fonctionnement du codeur et l'assigner à un Master, il faut introduire le codeur dans la Sanlist d'un Master.

L'illustration suivante montre la structure du bus avec le Master 1747-SDN d'un automate d'Allen-Bradley SLC500. Toutes les fenêtres qui suivent se réfèrent à cette configuration.

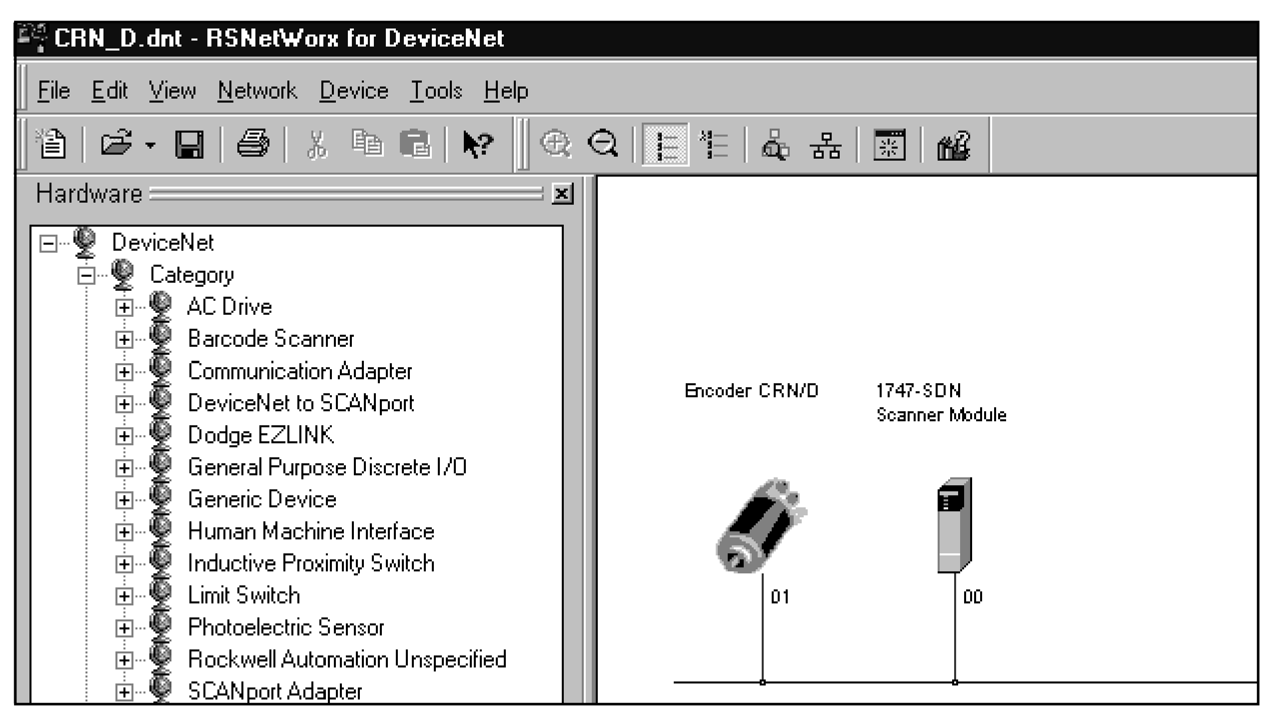


Illustration 10.6 Structure du bus avec Master

Cliquer deux fois sur le symbole du Scanner pour configurer les paramètres du Master. Dans le registre *Scanlist* se trouve à gauche une liste des appareils présents dans le bus. Sélectionner le codeur et le transférer, grâce à la touche fléchée, dans la liste de droite.

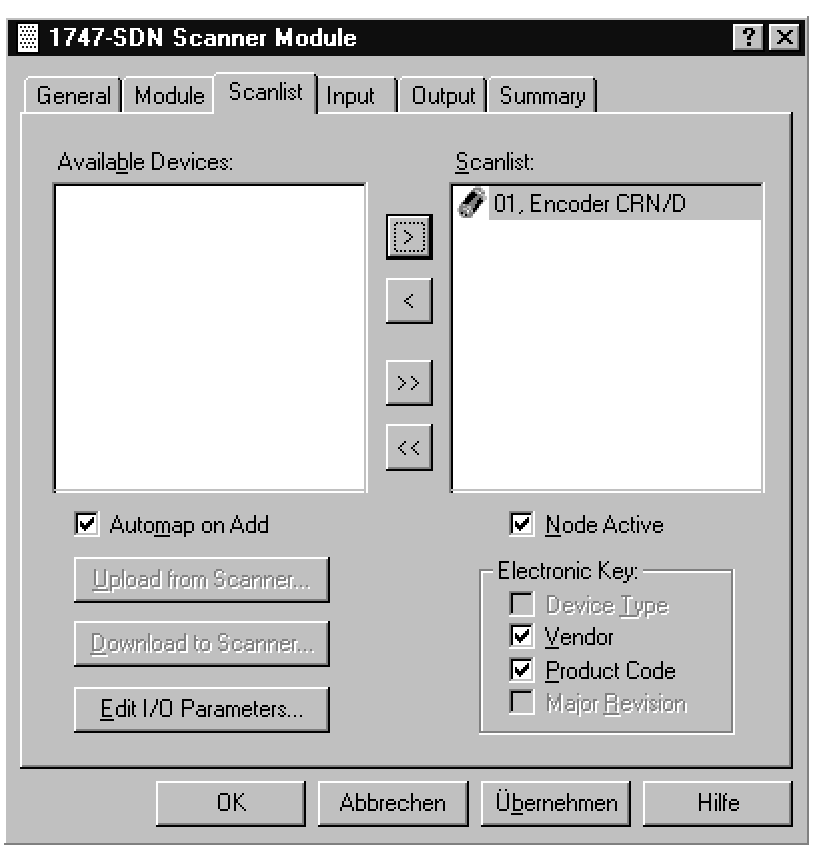


Illustration 10.7 Scanlist



Pour modifier le mode de fonctionnement du codeur, activer la touche *Edit I/O Parameters et* dans la fenêtre illustrée ci-dessous qui apparaît alors il est possible de choisir entre les différents modes de fonctionnement Poll-Mode, Bit-Strobe-Mode und Change of State / Cyclic-Mode. Il est également possible de choisir plusieurs modes de fonctionnement simultanément. Le nombre des données d'entrée (Rx-Size) doit toujours être 5 octets et celui des données de sortie 0 octet.

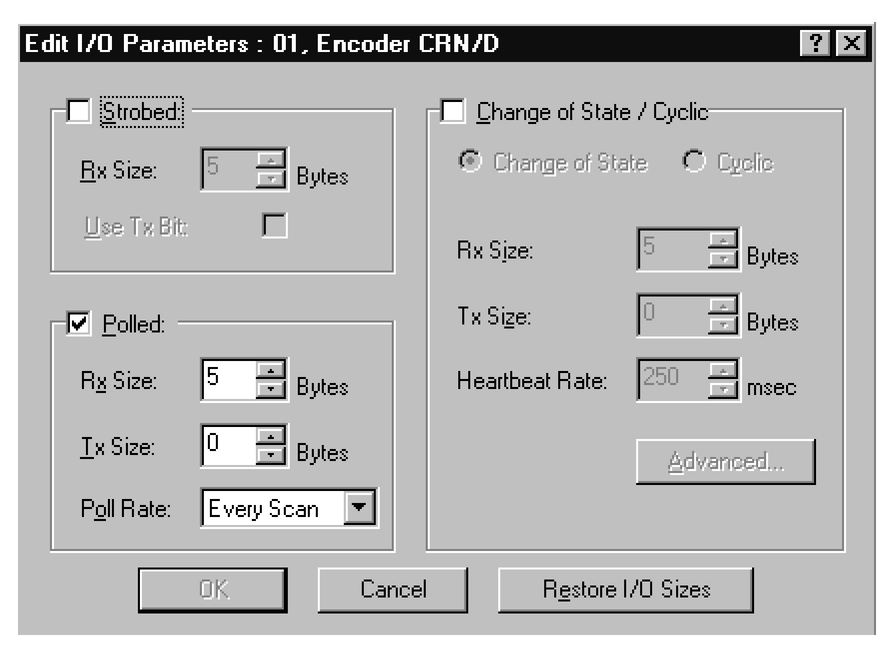


Illustration 10.8 Configuration du mode de fonctionnement du codeur

Le codeur affecté de son mode de fonctionnement souhaité est à présent dans la Scanlist du Master. Afin de pouvoir avoir accès aux données de l'automate, il faut indiquer au Scanner où se trouvent les données du codeur dans l'entrée de l'automate. Ceci a lieu grâce au **Mapping** qui peut être réalisé manuellement ou automatiquement.

Un Mapping a été réalisé automatiquement si, lors de l'introduction du codeur dans la Scanlist, le registre *Automap On Add* était activé (voir illustration Abb. 10.7). Dans *Input* le codeur doit apparaître comme dans l'illustration ci-dessous (dans cet exemple avec mode de fonctionnement Polling Mode uniquement).

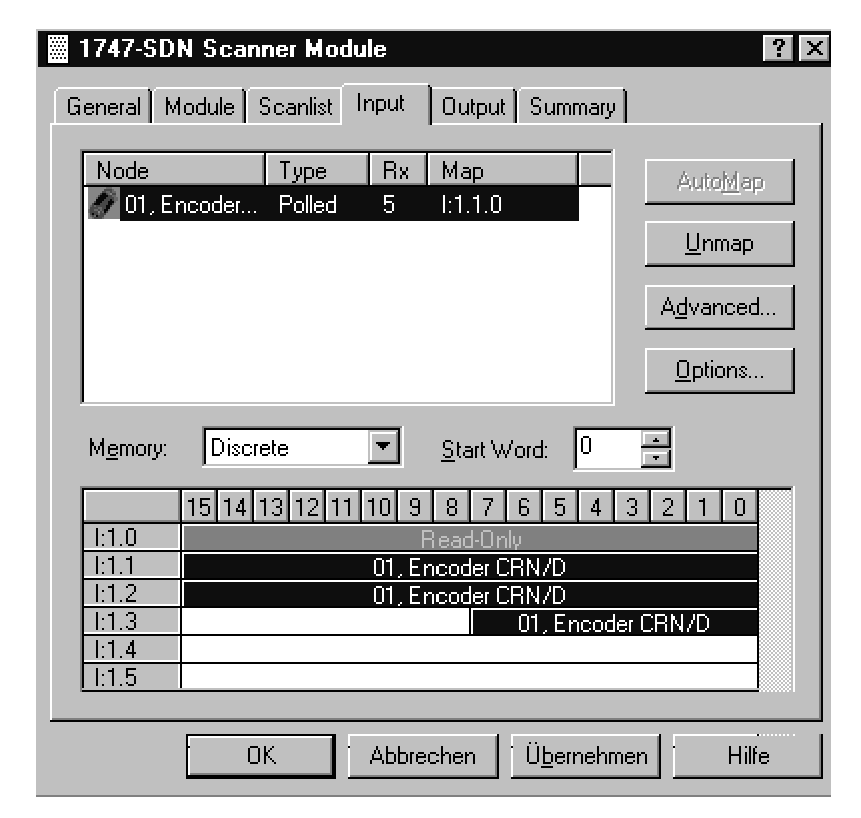


Illustration 10.9 Mapping



Quitter la fenêtre avec OK, enregistrer le projet et aller online. Après avoir laissé au RS-Networx le temps de lire la structure de bus, cliquer sous *Network* la rubrique *Download to Network* pour transmettre la structure complète du bus au Scanner.

10.5 Adressage et réglage de la vitesse de transmission d'un codeur avec connecteur

Le modèle avec connecteur CRN/D permet l'adressage (MAC-ID) et le réglage de la vitesse de transmission (125 kB, 250KB, 500KB) via Software.

Le réglage de la vitesse de transmission devrait toujours avoir lieu lors d'une connexion point par point avec le participant car la modification de la vitesse de transmission d'un participant du réseau peut anéantir tout le fonctionnement du bus. L'adressage d'un participant peut être, lui, effectué pendant le fonctionnement du bus. Faire attention cependant de ne pas utiliser une adresse déjà attribuée à un autre participant.

La modification de la vitesse de transmission n'est active qu'après une nouvelle mise sous tension on/off, la modification de l'adresse prend effet immédiatement.

Etablir une connexion point par point avec le codeur avec connecteur et choisir dans le RS-Networx sous *Tools* la rubrique *Node Commissioning*. Dans la fenêtre qui apparaît choisir le codeur à modifier en activant la touche *Browse*. Les *Current Device Settings* sont alors indiqués. Sous le registre *New Device Settings* il est possible d'entrer les nouvelles valeurs pour la vitesse de transmission et l'adresse du participant. L'exemple ci-dessous en est une possible illustration :

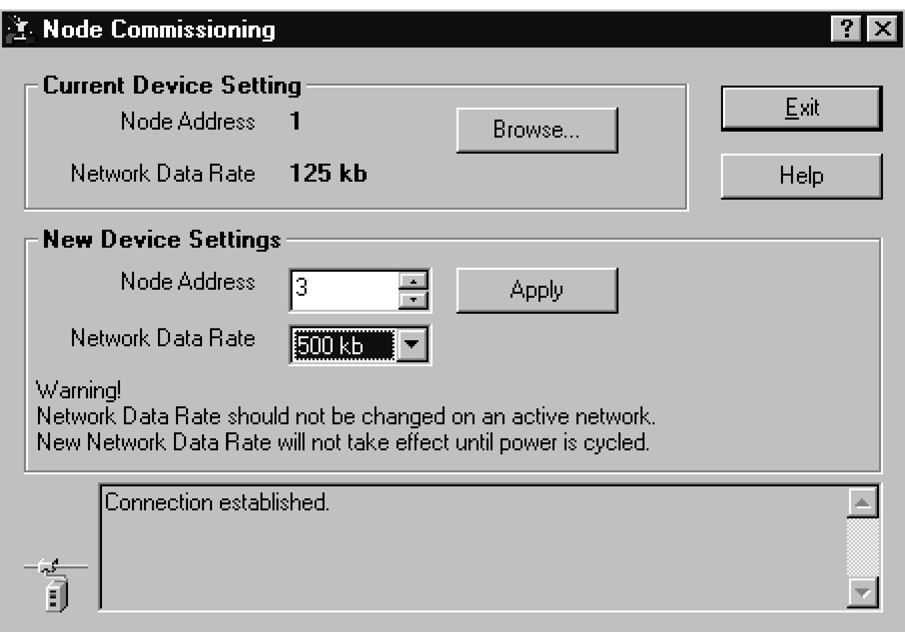


Illustration 10.10 Adressage et réglage de la vitesse de transmission

Activer la touche *Apply* pour que la modification de l'adresse et de la vitesse de transmission ait lieu et que les nouvelles valeurs apparaissent sous *Current Device Settings*.

10.6 Enregistrer les paramètres dans l'EEPROM

Cet enregistrement, pour parer à une coupure de secteur, est effectué via „Class Instance Editor".

Dans le bus marquer le codeur et, online, aller sous *Device* dans la rubrique *Class Instance Editor*. Dans le registre *Data Address,* entrer *Class* 0x23 (Position Sensor Object) et activer le service Save (0x16) dans *Service Code*. Activer enfin la touche *Send*. (voir illustration 10.11)

L'enregistrement des paramètres est confirmé dans la ligne d'état.



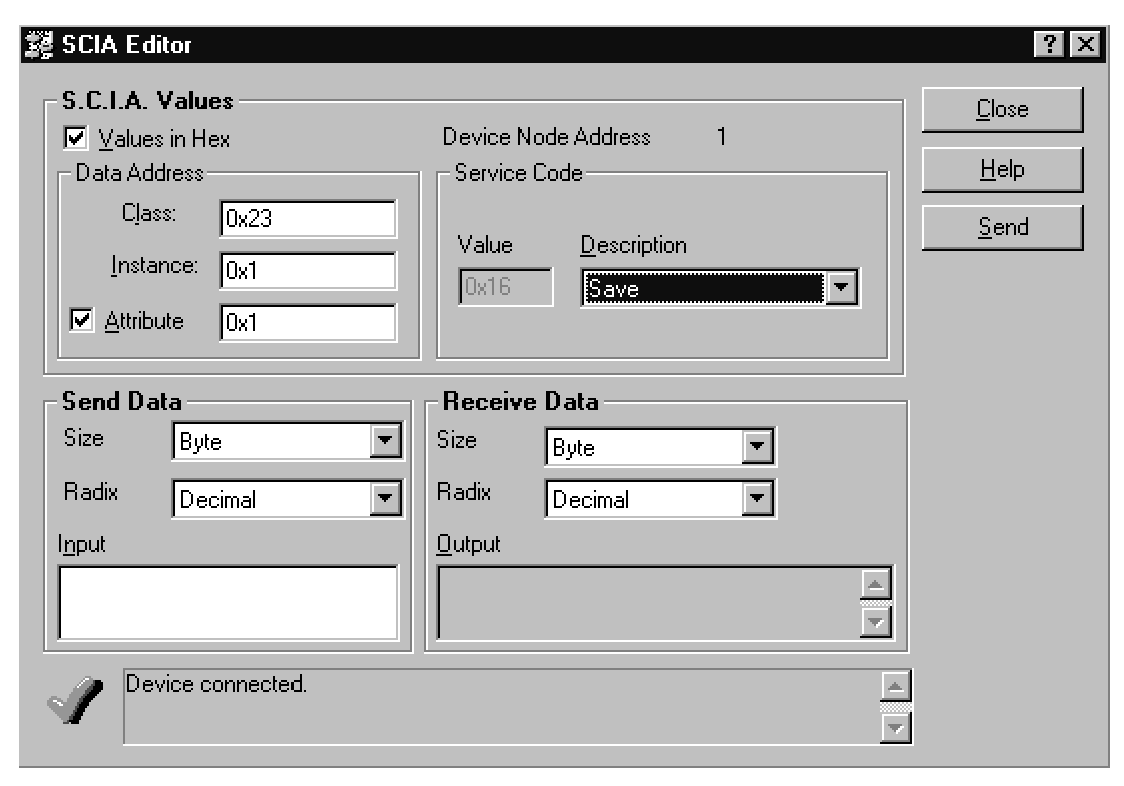


Illustration 10.11 Enregistrer les paramètres dans l'EEPROM

10.7 Charger les valeurs par défaut des paramètres

Il est possible de réinstaller toutes les valeurs par défaut des paramètres du codeur avec la commande „Restore" Service Code 21_{dez} du Position-Sensor-Object.

Marquer le codeur dans le bus et, online, aller sous *Device* dans la rubrique *Class Instance Editor*. Dans le registre *Data Address,* entrer *Class* 0x23 (Position Sensor Object) et activer le Service *Restore* (0x15) dans *Service Code*. Activer enfin la touche *Send*.

Les valeurs par défaut sont actives après avoir éteint et remis en route le codeur (tension on/off).

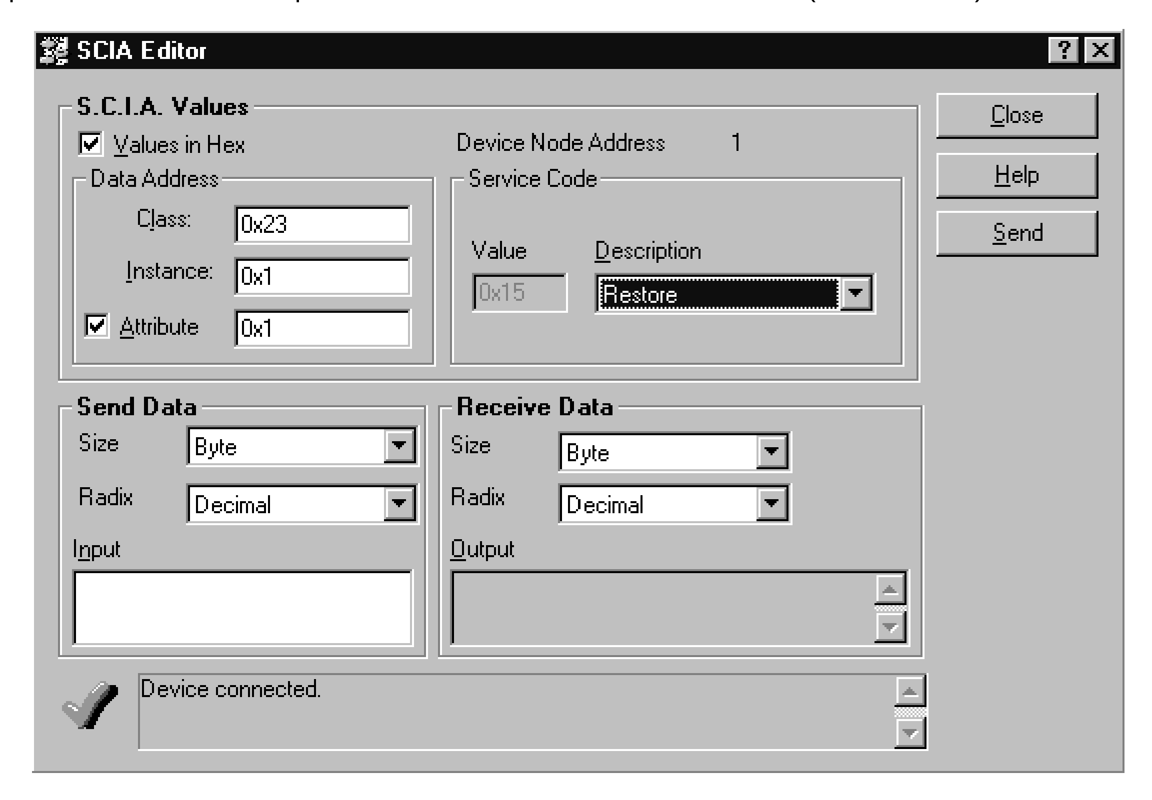


Illustration 10.12 Charger les valeurs par défaut des paramètres



10.8 Lire un attribut

Via le „Class Instance Editor" il est possible de lire tous les attributs (caractérisés comme lisibles) des divers DeviceNet-Objects présents dans le codeur. La liste des objets et des services attenants se trouve décrite dans le chapitre 8.

La lecture de tous les attributs s'effectue grâce au service **Get Attribute** (Service-ID 0xE), l'écriture grâce au service **Set Attribute** (Service-ID 0x10).

Dans l'exemple ci-dessous il s'agit de lire le numéro de série (Attribut 6 de l'Identity Object).

Sélectionner le codeur dans le bus et, online, aller sous *Device* dans la rubrique *Class Instance Editor*. Dans le registre *Data Address,* entrer *Class* 0x1 (Identity Object), et *Attribute* 0x6 (Seriennummer). Sous *Service Code* activer le service *Get Single Attribute* (0xE). Dans le registre *Received Data*, sous *Size* acitver Double et sous *Radix* Hexadecimal. Activer enfin la touche *Send*. Le numéro de série apparaît alors dans la fenêtre de restitution.

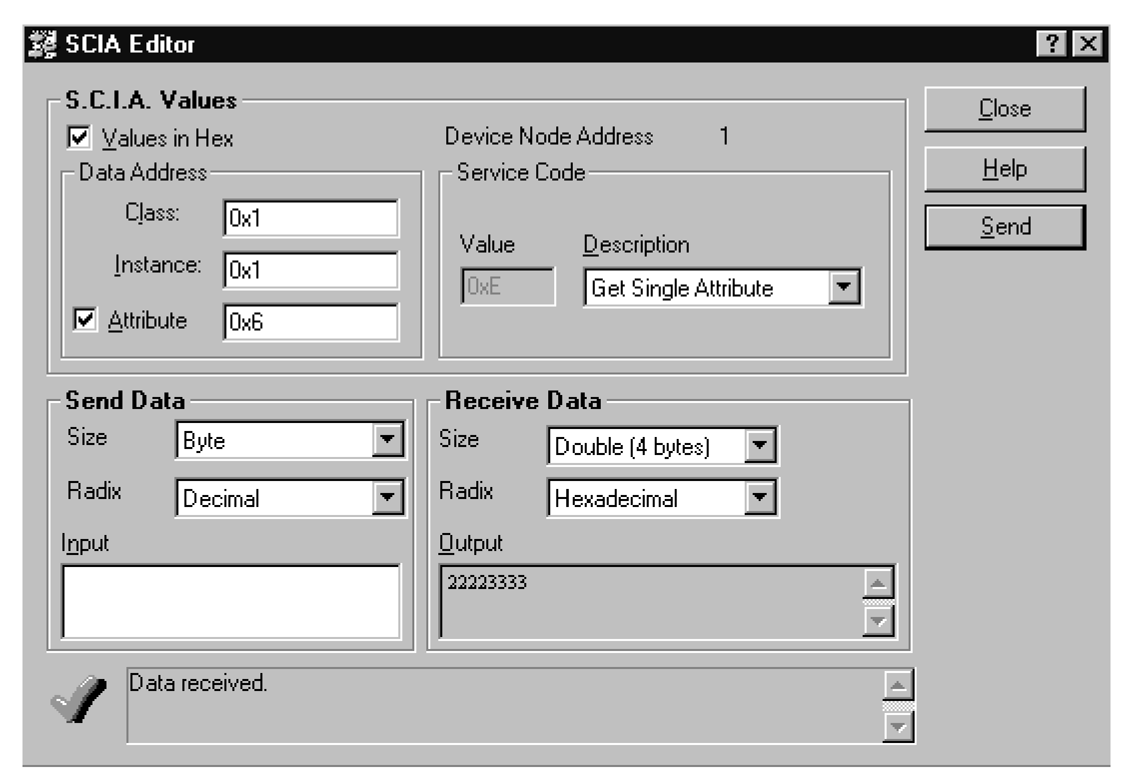


Illustration 10.13 Lire le numéro de série

Appendice A: Bibliographie

|1| ODVA - Open DeviceNet Vendor Association DeviceNet Specifications Release 2.0
|2| DeviceNet Manager Software user Manual 1787 - MGR